Títol: Optimització de la Base de Dades de Genexp

Volum: 1/1

Alumne: Gerard Muñoz Martínez

Director: Xavier Messeguer Peypoch

Departament: LSI

Data: 17 de juny del 2009

DADES DEL PROJECTE

Títol del Projecte: Optimització de la Base de Dades de Genexp

Nom de l'estudiant: Gerard Muñoz Martínez
Titulació: Enginyeria Informàtica
Crèdits: 37,5
Director: Xavier Messeguer Peypoch
Departament: LSI

MEMBRES DEL TRIBUNAL (nom i signatura)

President: ALBERTO ABELLO GAMAZO

Vocal: MANEL CANALES GABRIEL

Secretari: XAVIER MESSEGUER PEYPOCH

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

FACULTAT D'INFORMÀTICA DE BARCELONA
Universitat Politècnica de Catalunya

Optimització de la base de dades de l'explorador genòmic Genexp

---

Projecte de Final de Carrera
Enginyeria Informàtica

Alumne: Gerard Muñoz Martínez
Director: Xavier Messeguer Peypoch
Juny 2009

Agraïments

En primer lloc vull agrair a totes aquelles persones que, en major o menor mesura, han fet possible la realització d'aquest projecte.

Al Xavier Messeguer Peypoch, el Director d'aquest projecte, per oferir-me la possibilitat de realitzar-lo i pel seu continu suport al llarg del seu desenvolupament.

Al Bernat Gel Moreno, desenvolupador de Genexp, per la seva inestimable ajuda al llarg de la realització de tot el projecte.

A tots els membres del grup desenvolupador de Genomport per haver creat un ambient de treball agradable i on dóna gust poder col·laborar-hi.

A la meva companya, la Sílvia, que m'ha donat tot el seu suport, tant necessari en els llargs dies de feina.

A tota la meva família que sempre ha estat darrera meu i en especial als meus pares, el Jordi i la Isabel, sense els quals no podria haver realitzat aquesta carrera.

Índex

1 Introducció

Aquest document és la memòria del Projecte de Final de Carrera que tracta sobre la optimització de la base de dades de l'explorador genòmic Genexp i de la creació d'una eina de control de les seves dades.

1.1 Motivació del projecte

Quan ara fa un any vaig decidir-me per aquest projecte tenia molt clar que volia fer un projecte relacionat amb la bioinformàtica. Hi ha varis motius personals que em van portar a fer aquesta tria:

- La creença de que aquest és una disciplina molt recent, on encara hi ha molt per descobrir i experimentar i on et sents amb la satisfacció de ser pioner d'allò que fas.
- La convicció de que la informàtica té una forta implantació en molts camps, com per exemple l'enginyeria, el medi ambient, l'educació, la medicina, etc... Trobo molt reconfortant saber que amb la meva feina estic aportant el meu granet de sorra a la investigació per millorar la qualitat de vida de les persones que pateixen malalties ara per ara incurables.

A més a més aquest projecte tenia una forta component de disseny de bases de dades que era un àmbit de la carrera que sempre m'havia agradat i que tenia ganes de posar en pràctica. A més, en aquest cas, es tractava amb un gran volum de dades, fet que comportava un anàlisi molt detallat de les estructures usades per tal d'obtenir la màxima eficiència.

1.2 Presentació del projecte

En aquest apartat presentaré la temàtica del projecte i el situaré dins del context del projecte Genomport i de l'explorador genòmic Genexp.

Genomport

Genomport (*http://www.genomport.cat*) és un portal web que pretén donar resposta a la necessitat d'eines acadèmiques en l'àmbit de la biomedicina, donar suport a la recerca en aquest àmbit oferint eines que facilitin el treball dels investigadors i finalment convertir-se en un punt de trobada per a aquelles persones, com metges, malalts i investigador, sobre un tema concret. El portal ofereix tres entorns ben diferenciats:

- **3DPort:** L'entorn ofereix eines de visualització que permeten veure i manipular imatges del cos humà. Aquest entorn permet accedir a tres aplicacions:
  - De l'òrgan al DNA: Té dues funcionalitats ben diferenciades:
    - Zoom en profunditat: Aquesta aplicació permet fer un zoom a partir d'una successió de fotografies reals d'òrgans. Així doncs l'aplicació permet anar des de l'òrgan fins a la cèl·lula i posteriorment fins al nucli per arribar a distingir el DNA.

  - Visió circular: Permet fer rotacions sobre els òrgans de manera que podem observar-los des de tots els angles.
- L'esquelet en 3D: Aquesta aplicació permet representar en tres dimensions diferents parts del cos de les quals podem diferenciar convenientment les diferents textures: ossos, epitelis, cartílags... L'usuari pot interactuar amb el model 3D fent-hi rotacions i zooms.
- El meu cos: Avui en dia les diverses tècniques que permeten conèixer l'estat del nostre cos (tomografies, ressonàncies, ...) són a l'abast de molta gent. L'aplicació permet a l'usuari reconstruir en 3D el seu cos a partir de les imatges de les seves pròpies ressonàncies i TACs: de fet es tracta de l'anterior punt aplicat al cos de l'usuari.

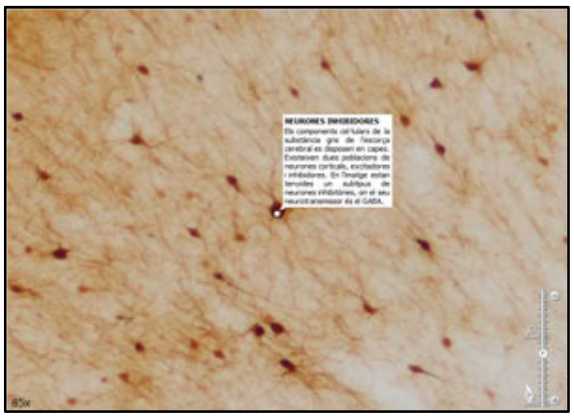


Fig. 1.1: Teixit cerebral a l'aplicació "De l'òrgan al DNA"

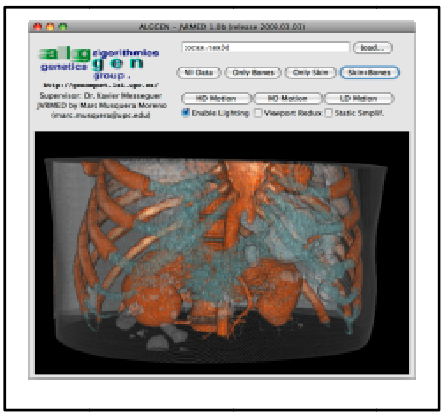


Fig. 1.2: Imatge tridimensional del tòrax a l'aplicació "L'esquelet en 3D"

- **GenPort:** En aquest entorn hi trobem el navegador genòmic Genexp que ens permet navegar per el genoma amb una interfície semblant a la de Google Maps. En el següent punt analitzarem més a fons aquest navegador.
- **Comport:** Aquest entorn ofereix un punt d'informació i comunicació al voltant dels ítems mostrats en el portal, com gens, SNPs, òrgans, malalties comunes, malalties gèniques, etc... Es permetrà als usuaris crear una comunitat al voltant de la seva malaltia on podrà explicar experiències, consells...

L'explorador genòmic Genexp

Genexp és un explorador genòmic que permet navegar per les seqüències d'ADN d'una manera agradable i intuïtiva. L'aplicació, que està essent elaborada per el doctorand Bernat Gel, permet obtenir informació biològica ràpidament i de forma visual.

Genexp és una aplicació web accessible per tots els investigadors a través dels seus navegadors. Aquesta universalitat en l'accés a l'aplicació té per força unes conseqüències en els seus requeriments:

- S'ha de poder executar en el navegador sense que faci falta instal·lar software addicional.
- S'ha d'executar de forma ràpida ja que sinó no serà agradable de fer servir.

- No pot fer un ús extensiu de memòria ja que s'ha d'executar en el navegador.
- Ha de mostrar dades reals, fiables i actualitzades.

L'aplicació permet seleccionar quin organisme i cromosoma volem veure. A continuació hem de seleccionar quins *track*s volem visualitzar d'aquest cromosoma. Un *track* és cada un dels tipus d'informació genòmica que volem mostrar, com per exemple la seqüència genòmica, els gens, el introns i els exons...

A la figura 1.3 observem Genexp mostrant una part del cromosoma 10 de l'Homo Sapiens del que es mostren 4 *tracks*: la seqüència, els contigs, els gens i els STS.

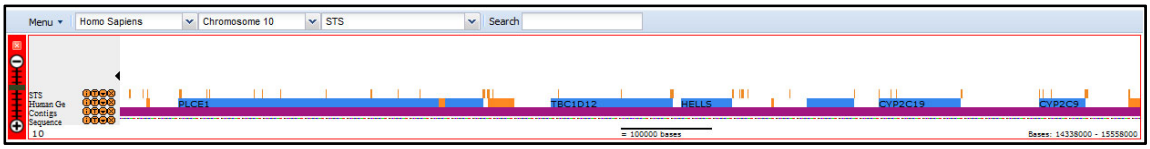


Fig 1.3 L'explorador genòmic Genexp

L'arquitectura de l'aplicació té dues parts ben diferenciades, la part servidor i la part client que descriuré breument a continuació:

- **Part client:** És la que s'executa en el navegador de l'usuari i ha estat implementada amb Javascript. Permet seleccionar quin organisme i cromosoma es vol visualitzar, així com escollir quina informació, o *Track*, es vol mostrar. També permet fer cerques i obtenir informació addicional sobre cada una de les característiques, o *features*, mostrades. És important notar que un cop les dades són mostrades tal i com es veu a la figura 1.3. Podem clicar en una de les zones i obtenir informació addicional d'aquella característica. L'aplicació es connecta a la part servidor usant les tecnologies AJAX i JSON.
- **Part servidor:** Dóna resposta a les peticions assíncrones del client. Es connecta a la base de dades i envia les dades a la part client. Els scripts que generen la resposta estan implementats en PHP i Perl.

Així doncs, la informació genòmica que es mostra en el navegador prové originalment d'una base de dades que es troba en el servidor. És imprescindible que es retorni la informació de forma ràpida ja que sinò l'aplicació no es comportaria d'una manera agradable. A continuació analitzarem una mica aquesta base de dades.

La base de dades de Genexp

La base de dades de Genexp ha de contenir tota la informació que es pot mostrar en el navegador genòmic. Cal notar que aquesta base de dades té un volum considerable ja que la quantitat d'informació genòmica disponible és molt elevada (per exemple només l'ésser humà ja té unes 3.000.000.000 bases, més de 20 milions de *SNP* o 35000 gens diferents). Aquesta base de dades pot anar creixent al llarg del temps doncs es poden anar introduint noves característiques o nous organismes, fet que multiplicaria les dades emmagatzemades. Aquest creixement no s'hauria de notar en els temps de resposta que haurien de restar baixos. Cal doncs que hi hagi unes estructures d'indexació eficients

que permetin accedir a les dades de forma ràpida. A més, cal que les dades estiguin organitzades de manera que l'aplicació pugui accedir-hi de forma eficient.

Quan vaig començar el projecte, Genexp ja tenia una base de dades implementada, però aquesta no era prou ràpida i feia que l'aplicació anés molt lenta. La meva tasca, doncs, ha estat redissenyar-la per tal de minimitzar els temps d'accés i fer-la més escalable.

És important notar d'on s'extreu la informació que es guarda en aquesta base de dades. Aquestes dades provenen de vàries bases de dades públiques que es troben a la xarxa (com veurem en el punt 2.2 de la memòria). Aquestes dades es troben en fitxers que tenen els seus propis formats que poden no tenir res a veure entre ells. A més, les diferents característiques poden estar en diferents bases de dades, per exemple, els gens els trobem a la base de dades Genbank mentre que els SNPs els trobem a la dbSNP, ambdues amb formats de fitxer diferents. Cal doncs que una aplicació les parsegi per a poder-les guardar a la base de dades.

Com s'ha dit el nombre de dades és considerable i es fa molt útil una aplicació que ens mostri quines dades tenim emmagatzemades a la base de dades. Així doncs, la meva tasca també ha estat desenvolupar aquesta aplicació, que té la funció d'administrar dades com quins organismes, cromosomes o característiques conté la base de dades però també descarregar la informació genòmica d'Internet, parsejar-la i guardar-la a la base de dades. És important notar que només s'ha fet un parser específic per la base de dades Genbank, que conté informació sobre gens, STS, contigs i Introns i Exons. Tot i que s'ha dissenyat l'aplicació de manera que pugui contenir més parsers es va considerar necessari centrar-se amb un sol parser i deixar el desenvolupament dels altres parsers fora de l'abast del projecte.

És rellevant també el fet de que la base de dades pot estar sotmesa a una alta càrrega de treball, sobretot pel que fa a la lectura d'informació. Per tant també s'ha hagut de dissenyar una estructura de servidors que facin possible aquesta càrrega de treball.

A mode de resum, els objectius del projecte són, doncs, els següents:

- Implementar una nova base de dades per a GenExp
- Dissenyar una aplicació que permeti gestionar les dades de la base de dades
- Definir una arquitectura de servidors

1.3 Organització de la memòria

En aquest punt explicaré com està estructurada aquesta memòria. A més d'aquest apartat introductori trobem els següents capítols:

Introducció a la bioinformàtica

En aquest punt es fa una aproximació als conceptes relacionats amb la bioinformàtica que el lector ha de saber per entendre les implicacions que té la naturalesa de les dades

genòmiques en el disseny de la base de dades. També es fa una breu descripció de les bases de dades genòmiques que hi ha disponibles a la xarxa.

Planificació del projecte

Aquí es fa la planificació del projecte, això inclou la definició d'objectius i abast, així com l'anàlisi de requeriments. També conté l'anàlisi de riscos, la definició d'activitats i l'estudi dels costos associats al desenvolupament del projecte.

Anàlisi del sistema original

En aquest apartat s'analitzen quins són els requeriments que té l'aplicació respecte a la base de dades. Això es fa analitzant el funcionament del navegador genòmic i veient quin és el tipus d'informació que requereix en cada moment. També es fa un anàlisi de la base de dades original per veure quin és el tipus d'informació que cal guardar, així com concloure quins són els punts febles que fan que no sigui funcional. En aquest punt també hi trobem els resultats dels diferents tests que s'han realitzat per obtenir els temps de resposta de la base de dades original.

Optimització de la base de dades

Aquí s'expliquen quines han estat les decisions que s'han pres en el disseny de la base de dades i també s'analitzaran els resultats obtinguts després de realitzar els diferents tests.

Desenvolupament de l'aplicació de control

En aquest punt es descriuran els aspectes més rellevants del procés de disseny de l'aplicació de control de les dades.

Conclusions

En aquest apartat es fa una valoració general del desenvolupament del projecte així com l'anàlisi d'acompliment dels objectius.

Fonts i referències

Aquí es descriu la llista de recursos documentals usats durant la realització del projecte.

Annex 1: Glossari de termes

Aquest annex recull la definició dels termes biològics que ha calgut tenir en compte per desenvolupar el projecte.

Annex 2: Ampliacions dels models conceptuals

En aquest annex hi trobem els models conceptuals en UML de la base de dades abans de ser optimitzada i després del procés d'optimització.

Annex 3: Descripció dels tipus de dades usats

Aquest annex recull la relació dels tipus de dades usats en la base de dades optimitzada.

Annex 4: Sentències SQL

En aquest annex hi trobem les sentències SQL de creació de les taules de la base de dades optimitzada.

Annex 5: Manual d'usuari de l'aplicació de control

Aquí hi trobem el manual d'usuari de l'aplicació de control de les dades.

2 Introducció a la bioinformàtica

En aquest apartat es farà una breu aproximació als conceptes biològics que cal conèixer per comprendre els aspectes biològics del projecte.

2.1 El genoma

Des de que Oswald Theodore Avery, Colin McLeod i Maclyn McCarty van demostrar el 1944 que el DNA es tractava del material genètic que defineix els nostres trets diferencials, la biologia ha pres una empenta infrenable i ha obert nous horitzons i la possibilitat de curar malalties fins ara incurables. Aquests fets han fet de l'estudi de la genètica un centre d'interès important dins del panorama científic actual. La genètica és una ciència molt complexa que tracta de comprendre com l'herència biològica és transmesa d'una generació a la següent i com s'efectua el desenvolupament de les característiques que controlen aquests processos.

Fem ara una aproximació als conceptes més rellevants de la genètica:

És sabut que els cossos dels organismes estan formats per ossos, músculs, teixits... al seu temps aquestes estructures estan formades per cèl·lules de diferents tipus, cadascuna d'elles amb diferents funcions i característiques. No obstant, totes les cèl·lules d'un organisme tenen un element comú anomenat codi genètic (o DNA) que és el mateix en totes elles. Aquest codi acostuma estar organitzat en el que anomenem cromosomes i no és res més que un conjunt de grans molècules formades per una successió de quatre elements diferents, quatre bases nitrogenades, que fem correspondre amb les lletres A, C, G i T. Així doncs la informació genètica està estructurada en cromosomes. Però com obtenim aquesta informació? Gràcies a les tècniques de seqüenciació, que utilitzen mètodes químics i computacionals, podem obtenir la tira de bases que formen el codi genètic d'un cromosoma. Malauradament a vegades només obtenim trossos de codi genètic però amb parts de codi desconegut entremig. Cada un dels trossos coneguts és el que anomenem *contigs* o *locus*. En aquests *contigs* és on trobem la resta de característiques, com per exemple els gens, els quals estan repartits per tota la cadena de DNA, i cada un d'ells codifica una proteïna. Les proteïnes exerceixen un paper fonamental en els èssers vius i són les biomolècules més versàtils i diverses. Realitzen una enorme quantitat de funcions diferents en els organismes. Per tant, si un gen codifica de forma incorrecte una proteïna això pot repercutir en el mal funcionament de l'organisme. Aquesta és, de fet, la causa de moltes malalties.

Cal notar que no tot el gen intervé en la codificació de la proteïna, només un petit conjunt de les seves bases és utilitzat en aquest procés. Aquest conjunt de bases és el que anomenem *exons*. Els conjunts de bases que queden entre els diferents *exons* són els *introns*. També existeixen els anomenats *UTR* que són trossos de codi que marquen el inici (*UTR 5'*) i el final (*UTR 3'*) de la seqüència codificant. Podem veure-ho gràficament a la figura 2.1. Així doncs l'estudi dels *introns* i els *exons* també pren molta rellevància en l'estudi genètic.

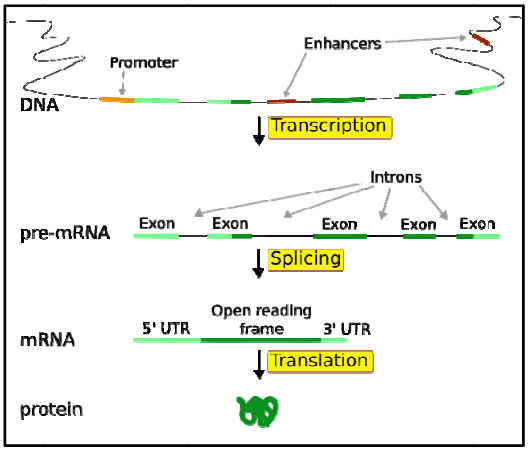


Fig. 2.1: Procés de codificació d'una proteïna

Un altre concepte rellevant són els *STS* (*Sequence-Tagged Sites*) que són trossos de seqüència genètica única dels quals es coneix la seva posició. També cal destacar l'existència dels *SNP* (*Single Nucleotide Polymorphisms*) que són bases identificades que varien entre diferents individus, aquests són, de fet, els responsables de les diferències entre els diferents individus d'una mateixa espècie.

A la següent figura, generada amb el navegador Genexp, podem observar com es representen aquests elements en diferents files, o *tracks*.

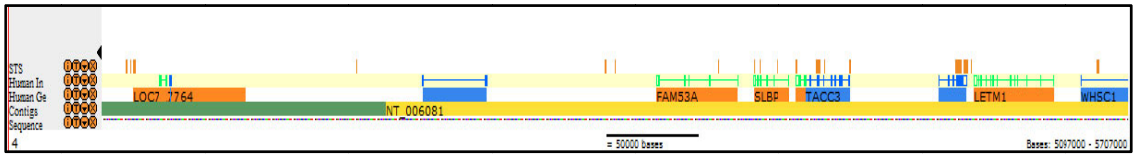


Fig. 2.2: Part del Cromosoma 4 de l'Homo Sapiens mostrada per Genexp

En la de més avall podem observar (o intuir) la seqüència de bases A,C,G,T que formen la tira de DNA del cromosoma. Per sobre d'aquesta identifiquem dos *contigs*. Cal notar que en aquesta imatge els *contigs* estan junts, sense espai desconegut pel mig. Això és degut a que tractem amb un model genètic simplificat on considerem que els *contigs* estan junts i no hi ha espai entre ells. Dins de cada *contig* observem els gens, identificats amb noms. Els gens tenen una posició d'inici i una de final, a més tenen el que anomenem *Strand* que indica el sentit en el que s'han de llegir, això és mostrat en el navegador en l'ús de colors diferents en funció de si tenen *Strand* positiu o *Strand* negatiu. Per sobre de cada gen observem una tira d'exons i introns. Els *exons* són els rectangles opacs i els *introns* són les línies que els uneixen. També es poden identificar uns rectangles buits al inici i final d'algunes de les seqüències codificants, es tracten dels *UTR*. Ja només ens queden per veure els *STS* que són els rectangles que hi ha a la línia superior i que identifiquen diferents seqüències etiquetades.

2.2 Bases de dades genòmiques

Durant la realització del projecte he tractat amb algunes bases de dades genòmiques. Les més importants són les que ofereix el NCBI (*National Center for Biotechnology Information*) que depèn del *National Institutes of Health* dels Estats Units. Les bases de dades del NCBI han esdevingut al llarg dels anys la referència pel que fa a informació genòmica.

Genbank

Una d'aquestes bases de dades és *GenBank* que conté la informació de les seqüències genètiques de varis organismes, entre ells l'ésser humà. GenBank no integra només dades del NCBI sinó que també conté informació provinent del DDBJ (*DNA DataBank of Japan*) i del EMBL (*European Molecular Biology Laboratory*), aquesta informació prové dels laboratoris de més de 100.000 organismes d'arreu del món. Aquest fet fa que la informació genòmica es dobli cada 18 mesos. A més a més la informació s'actualitza cada dos mesos, és per això que cal que la informació que mostri el explorador genòmic estigui actualitzada. A la següent figura veiem aquest creixement de forma gràfica.

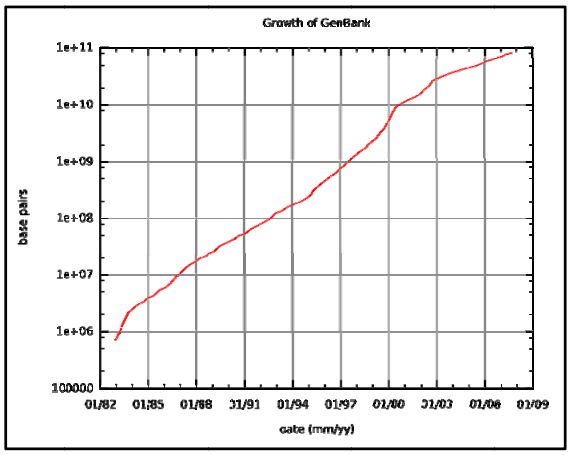


Fig. 2.3: Creixement del nombre de bases existents a Genbank.
Font: GenBank release notes from October, 2007

La informació de Genbank és accessible de diferents maneres. Podem usar el servei web anomenat *entrez*, que permet fer crides amb paràmetres específics i ens retorna un XML amb la informació genòmica. No obstant per a tractar amb volums grans de dades és millor treballar amb el servei ftp que ofereix, on la informació genòmica es troba organitzada en directoris per als diferents organismes. A través d'aquests directoris accedim als diferents cromosomes del organisme, els quals també conformen nous directoris. Dins d'aquests directoris ja hi trobem el fitxer amb la informació genètica. Són els anomenats fitxers gbk (amb la seqüència genètica del cromosoma) o gbs (sense la seqüència genètica).

Un exemple d'adreça on trobar aquests fitxers seria la següent:

```
ftp://ftp.ncbi.nih.gov/genomes/H_sapiens/CHR_04/hs_ref_chr4.gbs.gz
```

Es poden identificar els directoris d'organisme (H_sapiens) i de cromosoma (CHR_04). També es pot veure com el fitxer es troba comprimit, per tant caldrà descomprimir-lo per a poder-lo parsejar.

Analitzem ara quin és el format del fitxer Genbank. El format Genbank és un format bastant simple que consisteix en una seqüència de contigs, o *locus*. A continuació de la descripció de cada *locus* trobem la seqüència de característiques, siguin gens, sts, rnaHairpin... En aquestes característiques trobem la seva informació associada com el símbol, l'identificador, la posició, la longitud... Al seu torn a continuació de cada gen trobem la informació referent a la seva seqüència codificant (exons i introns). Així doncs, el format Genbank és un format relativament fàcil de parsejar ja que es podria generar amb una gramàtica bastant simple. Tot i així, i com veurem més endavant, s'ha utilitzat la llibreria *biojava* per a parsejar el fitxer.

3 Planificació del projecte

En aquest apartat es descriu quina la planificació del projecte així com la seva estimació de costos. També es definiran els objectius i l'abast del projecte.

3.1 Definició dels objectius

Com hem vist Genexp genera les imatges que mostra a partir de la informació emmagatzemada en una base de dades. Originalment Genexp ja disposava d'una base de dades però aquesta era molt lenta i a més s'havia arribat al límit de la seva capacitat, fet que feia necessari un redisseny de la mateixa per poder permetre que el navegador genòmic es comportés de manera fluïda.

El meu projecte consisteix en redissenyar i implementar de nou aquesta base de dades per tal de minimitzar els temps d'accés i fer-la més escalable. D'aquesta manera seria possible que l'aplicació Genexp tingués un comportament còmode i agradable.

Quan parlem d'informació biològica el nombre de dades amb el que hem de tractar és molt elevat. És per això que hem de tenir eines que ens ajudin a tractar aquestes dades i ens permetin gestionar-les. El projecte també consisteix en crear una aplicació per a gestionar les dades de la base de dades de Genexp i que ens permeti l'actualització de les mateixes de forma còmode.

Així doncs, els objectius del projecte són:

- **Implementar una nova base de dades per a GenExp**: Cal redissenyar la base de dades existent per augmentar el seu rendiment i la seva escalabilitat.
- **Dissenyar una aplicació que permeti gestionar les dades de la base de dades:** Cal que aquesta aplicació permeti tractar amb tota mena de dades biològiques que posteriorment es vulguin mostrar en el navegador genòmic. Per tant és necessari que sigui una aplicació escalable que ens permeti ampliar les seves funcionalitats en un futur.
- **Definir una arquitectura de servidors:** S'haurà de definir una estructura de servidors de base de dades per tal de minimitzar els temps d'accés i fer possible suportar una càrrega elevada del sistema.

3.2 Limitació de l'abast

A continuació delimitarem quins aspectes queden dins de l'abast del projecte:

Base de dades

- Es dissenyarà la base de dades de Genexp de manera que permeti la futura introducció de nous tipus de dades genòmiques. També s'estructurarà de manera que els temps d'accés siguin acceptables en quant a la fluïdesa de l'aplicació Genexp.
- S'implementarà la base de dades dissenyada en el SGBD que es consideri més convenient. També es crearan les estructures d'indexació necessàries per a minimitzar els temps d'accés.

- Es crearan jocs de proves i es farà una càrrega de dades amb dades reals o fictícies (depenent de la disponibilitat) per provar l'adequació dels temps d'accés.
- Es definirà una estructura de servidors que augmentin la robustesa i la seguretat i permetin una càrrega elevada del sistema.

Aplicació de control

- Es dissenyarà l'aplicació de control de les dades de Genexp, aquesta aplicació ha de poder tractar amb diferents fonts d'informació.
- S'implementarà l'aplicació de gestió general de manera que pugui gestionar els organismes i els seus cromosomes i cadascun dels seus diferents tracks. Cal que es puguin crear, modificar i eliminar.
- S'implementarà el parser específic de Genbank, això implica que s'han de poder tractar i introduir a la base de dades els contigs, els gens, els STS i les seqüències codificants (UTR, Exons i Introns). Cal que aquesta informació es descarregui de la base de dades de Genbank, es descomprimeixi, es parsegi i es guardi a la base de dades. Cal que el sistema monitoritzi el progrés d'aquestes operacions.
- L'aplicació ha de permetre gestionar diferents bases de dades per a poder tractar amb diferents versions de les dades.
- L'aplicació ha de tenir una interfície gràfica intuïtiva i fàcil d'utilitzar.

3.3 Anàlisi de requeriments

3.3.1 Requeriments de la base de dades

La base de dades té els següents requeriments tant de rendiment com de configuració. La definició de les dades que s'han de poder emmagatzemar es mostren en el model conceptual, descrit en el punt 5.1.1 de la memòria.

1. Ha de minimitzar els temps d'accés en dos tipus diferents de consultes:
   1.1. Consulta per obtenir totes les característiques d'un determinat track que es troben dins d'un rang de posicions específic.
   1.2. Consulta per obtenir informació addicional d'una característica determinada.
2. Els temps de resposta de la base de dades no seran superiors a 5 segons en el cas de les FeatureViews o de 10 segons en el cas de les ImageViews.
3. El SGBD ha de poder administrar eficientment una quantitat de dades d'1 TB aproximadament.
4. El SGBD ha d'estar desenvolupat en codi obert
5. Ha de mantenir la consistència de les dades.

3.3.2 Requeriments de l'aplicació de control de les dades

Requeriments funcionals

1. Aplicació principal
   1.1. Gestió d'organismes
      a) Ha de permetre crear els organismes
      b) Ha de permetre eliminar els organismes
      c) Ha de permetre modificar la informació dels organismes existents.
   1.2. Gestió de cromosomes
      a) Ha de permetre crear cromosomes dels organismes existents
      b) Ha de permetre eliminar cromosomes
      c) Ha de permetre modificar els cromosomes existents
   1.3. Gestió de tracks
      a) Ha de permetre crear els tracks sobre els organismes existents
      b) Ha de permetre eliminar els tracks
      c) Ha de permetre modificar la informació dels tracks existents
   1.4. Gestió de les diferents bases de dades
      a) Ha de permetre seleccionar una base de dades per a modificar-la
      b) Ha de permetre mantenir una versió congelada de la base de dades amb una versió antiga d'aquesta
      c) Ha de permetre mantenir una versió de còpia de la base de dades
      d) Ha de permetre configurar la informació (host, username, password) de les diferents bases de dades
2. Sub-aplicació del parser de Genbank
   2.1. Gestió de les dades existents
      a) Ha de mostrar de quins cromosomes hi ha dades genòmiques a la base de dades. Així com en quina versió es troben
      b) Ha de permetre actualitzar els cromosomes que es seleccionin
         - Ha de descarregar-se el fitxer gbs de la base de dades de Genbank, descomprimir-lo, parsejar-lo i introduir les dades a la base de dades, i a continuació esborrar-lo.
         - Els tipus de característiques que s'han de poder introduir a la base de dades a partir dels fitxers de Genbank són: contigs, gens, sts i seqüències codificants (utr, introns i exons)
         - Ha de monitoritzar el desenvolupament d'aquestes operacions mitjançant barres de progrés.
         - El procés no s'ha de poder aturar fins que s'hagi acabat d'actualitzar els cromosomes
   2.2. Gestió de la configuració dels organismes de Genbank
      a) Ha de poder introduir la URL del fitxer de Genbank que conté el número de versió actual de l'organisme
   2.3. Gestió de la configuració dels cromosomes de Genbank
      a) Ha de poder introduir la URL del fitxer gbs del cromosoma
   2.4. Gestió de la configuració dels tracks

a) Ha de permetre associar els tracks existents als diferents tipus de característiques que ofereixen els fitxers de Genbank.

Requeriments no funcionals

1. L'aplicació s'ha de poder executar en Windows, Linux i Solaris
2. L'aplicació ha de tenir una interfície intuïtiva i fàcil de fer servir
3. L'aplicació ha de ser robusta i escalable per poder afegir noves sub-aplicacions

3.4 Definició de les activitats

Les activitats a realitzar per a desenvolupar el projecte són:

Familiarització amb el sistema actual

1. **Anàlisi del model conceptual actual:** Donat que no es té documentació del model conceptual implementat caldrà fer enginyeria inversa a partir del model de dades implementat en el SGBD
2. **Anàlisi de la base de dades actual:** Anàlisi de totes les estructures de la base de dades com índexos, claus externes, tipus de dades...
3. **Identificació de les crides més habituals:** Cal cercar en el codi de Genexp quines són les crides que es realitzen per identificar quines són les necessitats de l'aplicació de cara a la base de dades.

Avaluació del sistema actual

1. **Programació d'scripts per avaluar el rendiment de la BD:** Cal implementar scripts que executin les crides identificades en l'etapa anterior per a poder obtenir resultats sobre el rendiment.
2. **Execució dels scripts i avaluació dels resultats:** Cal executar els scripts programats i obtenir resultats que podrem comparar amb els de la base de dades redissenyada.
3. **Identificació dels colls d'ampolla:** Amb els resultats obtinguts podem treure conclusions sobre quines són les crides que esdevenen colls d'ampolla de l'aplicació.

Disseny de la nova base de dades

1. **Disseny conceptual del model de dades:** S'han d'identificar quines són les diferents entitats o classes que formen el model conceptual així com les seves relacions.
2. **Disseny lògic de la base de dades:** Cal adaptar el model conceptual al model relacional.
3. **Tria del Sistema Gestor de Bases de Dades:** S'ha de triar el SGBD que millor s'adapti als requeriments especificats.
4. **Quantificació del volum de dades:** Caldrà quantificar les dades que volem tractar.
5. **Disseny físic de la base de dades:** Adaptar el model lògic al sistema gestor de bases de dades escollit.

6. **Definició final del model de la base de dades:** A partir del disseny físic i dels aspectes rellevants detectats en aquests dos últims apartats caldrà definir un model final de dades.

Implementació de la base de dades

1. **Implementació de la base de dades:** Cal implementar la nova base de dades en el SGBD, és a dir, cal crear les taules i les estructures d'accés.

Avaluació del sistema modificat

1. **Programació d'scripts per avaluar el rendiment de la BD:** Cal reprogramar els scripts de testejat per a que s'adaptin a la nova base de dades.
2. **Introducció de les dades existents en el sistema original:** Per a poder obtenir una comparació fiable caldrà introduir a la base de dades les mateixes dades existents en el sistema original.
3. **Execució dels scripts:** Cal executar els scripts programats i obtenir els resultats de cara a poder comparar-los amb la versió original.
4. **Introducció de les dades esperades:** Cal generar dades per introduir a la base de dades per a simular la càrrega esperada de la base de dades.
5. **Execució dels scripts:** Cal executar els scripts programats i obtenir els resultats de cara a analitzar si el sistema es comporta adequadament quan el volum de dades és l'esperat.
6. **Valoració dels resultats:** S'han d'analitzar els resultats per veure si són acceptables en quan a temps de resposta.

Disseny de l'aplicació de control de les dades

1. **Definició dels casos d'ús de l'aplicació:** A partir dels requeriments de l'aplicació cal definir els seus casos d'ús.
2. **Disseny de l'arquitectura de l'aplicació:** Cal definir quina serà l'arquitectura de l'aplicació.
3. **Tria d'entorn i tecnologies usades:** Cal triar un llenguatge de programació així com considerar altres aspectes físics de l'aplicació.
4. **Descripció dels casos d'ús:** Cal definir el comportament de l'aplicació en els diferents casos d'ús.

Implementació de l'aplicació de control

1. **Implementació de l'aplicació general:** Cal implementar les funcionalitats que formen part de l'aplicació general.
2. **Implementació de la sub-aplicació de Genbank:** Cal implementar les funcionalitats que formen part de la sub-aplicació de Genbank.

Testejat de l'aplicació de control

1. **Proves de les funcionalitats de l'aplicació de control:** Cal comprovar que es satisfacin els requeriments.

Realització de la documentació

1. **Desenvolupament de la documentació:** Cal desenvolupar documentació de les decisions preses en el desenvolupament.

3.5 Planificació de les activitats

La planificació del projecte ens permet fer una estimació del temps que comportarà realitzar-lo així com fer una estimació del seus costos associats. Cal considerar aquesta planificació com quelcom dinàmic que cal anar revisant i modificant en funció dels desviaments que es vagin produint. L'estimació de la duració de les tasques s'ha fet en base a l'experiència adquirida al llarg del desenvolupament d'activitats durant la carrera.

Diagrama de Gantt

El diagrama de Gantt associat a la planificació del projecte és el següent:

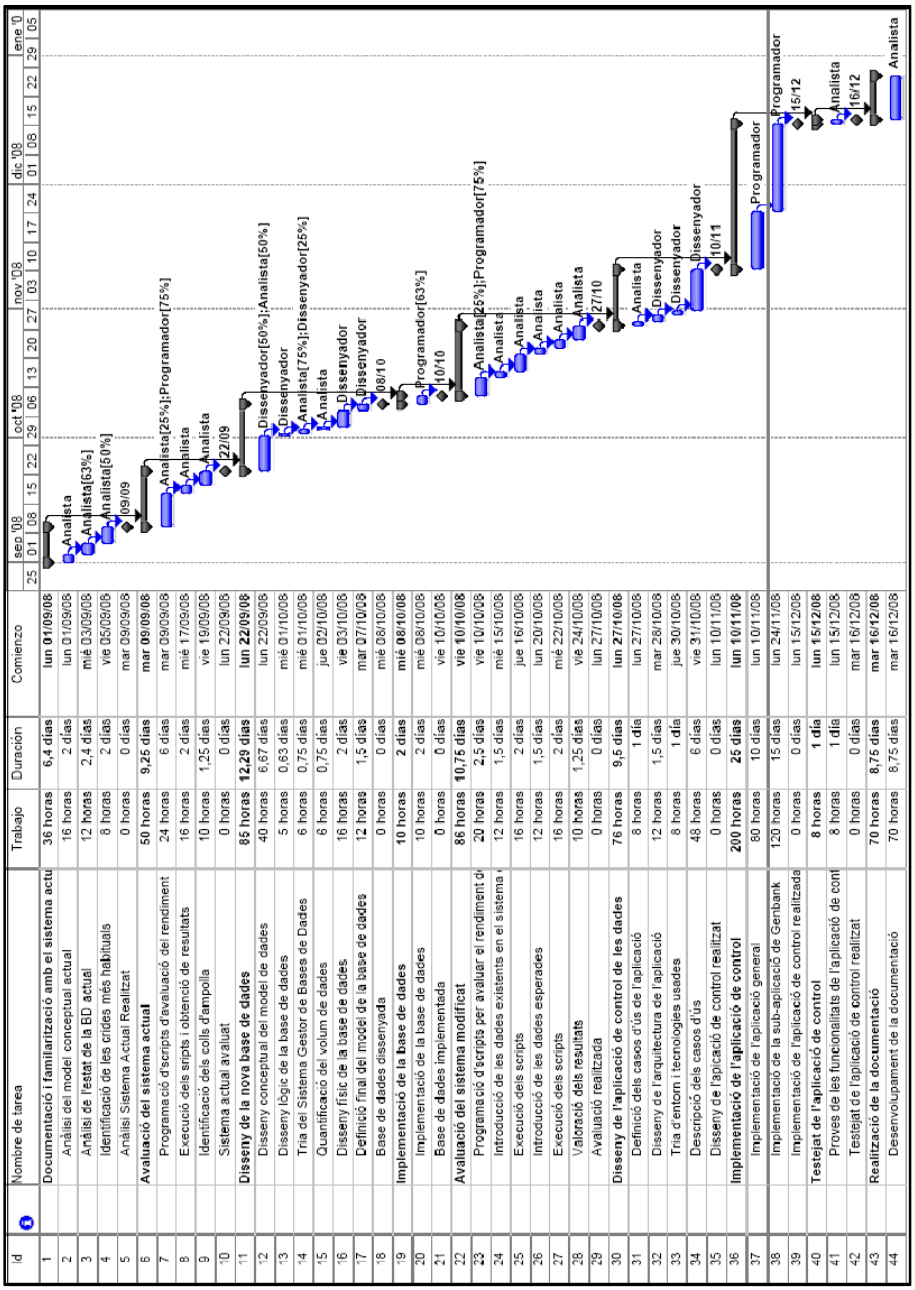


Fig. 3.1: Diagrama de Gantt del projecte

3.6 Anàlisi de costos

Un cop hem fet una planificació del treball que comporta cada tasca podem fer una estimació de costos del projecte. Per a això cal considerar els recursos tant humans com físics emprats en la realització del projecte.

3.6.1 Recursos humans del projecte

La partida en la que es destinen més recursos econòmics és en la de personal. Aquest projecte, com resulta obvi al ser un projecte final de carrera, no té un equip de personal especialitzat en cadascuna de les tasques, sinó que hi ha una única persona, el projectista, que s'encarrega de dur a terme totes elles, a part del Director del projecte que, per descomptat, ajuda en la realització del mateix.

Per mostrar un càlcul de costos destinat als recursos humans més realista es simularà un conjunt de personal que s'encarregaran de dur a terme les tasques explicitades en el diagrama de Gantt.

En el desenvolupament d'un projecte informàtic intervenen diferents rols, el cap de projecte, l'analista, el dissenyador i el programador. Cadascun d'aquests rols es fa càrrec d'una part del projecte segons les seves competències. En aquest projecte es suposarà que la figura del cap de projecte està duta a terme per l'analista, que a més a més, és l'encarregat de la part d'anàlisi i especificació de requisits així com de la validació dels resultats. El dissenyador és l'encarregat de fer el disseny del sistema software i el programador és l'encarregat de la codificar-lo.

A la taula 3.1 es mostren el preu per hora fixat per a cada escala laboral.

Nom del recurs	Tarifa
Analista	60 €/h
Dissenyador	45 €/h
Programador	20 €/h

Taula 3.1: Tarifes dels recursos

A partir del calendari del projecte, l'assignació de recursos humans a cada tasca i el salari per a cada treballador és possible realitzar el càlcul del cost total del projecte que es pot veure a la taula 3.2.

Nom de la tasca	Recurs personal	Hores de feina	Cost total
Familiarització amb el sistema actual			
Anàlisi del model conceptual actual	Analista	16 hores	960,00 €
Anàlisi de la base de dades actual	Analista	12 hores	720,00 €
Identificació de les crides més habituals	Analista	8 hores	480,00 €
Avaluació del sistema actual			
Programació d'scripts per avaluar el rendiment de la BD	Analista Programador	24 hores	960,00 €
Execució dels scripts i avaluació dels resultats	Analista	16 hores	960,00 €
Identificació dels colls d'ampolla	Analista	10 hores	600,00 €

Nom de la tasca	Recurs personal	Hores de feina	Cost total
Disseny de la nova base de dades			
Disseny conceptual del model de dades	Analista Dissenyador	24 hores	2.000,00€
Disseny lògic de la base de dades	Dissenyador	5 hores	225,00€
Tria del Sistema Gestor de Bases de Dades	Analista Dissenyador	6 hores	337,50 €
Disseny fisic de la base de dades	Dissenyador	16 hores	720,00 €
Quantificació del volum de dades	Analista	6 hores	360,00 €
Definició final del model de la base de dades	Dissenyador	12 hores	540,00 €
Implementació de la base de dades			
Implementació de la base de dades	Programador	10 hores	200,00 €
Avaluació del sistema modificat			
Programació d'scripts per avaluar el rendiment de la BD	Analista Programador	20 hores	600,00 €
Introducció de les dades existents en el sistema original	Analista	12 hores	720,00 €
Execució dels scripts	Analista	16 hores	960,00 €
Introducció de les dades esperades	Analista	12 hores	720,00 €
Execució dels scripts	Analista	16 hores	960,00 €
Valoració dels resultats	Analista	10 hores	600,00 €
Disseny de l'aplicació del control de dades			
Definició dels casos d'ús de l'aplicació	Analista	8 hores	480,00 €
Disseny de l'arquitectura de l'aplicació	Dissenyador	12 hores	540,00 €
Tria d'entorn i tecnologies usades	Dissenyador	8 hores	360,00 €
Descripció dels casos d'ús	Dissenyador	48 hores	2.160,00 €
Implementació de l'aplicació de control			
Implementació de l'aplicació general	Programador	80 hores	1.600,00 €
Implementació de la sub-aplicació de Genbank	Programador	120 hores	2.400,00 €
Testejat de l'aplicació de control			
Proves de funcionalitats de l'aplicació de control	Analista	8 hores	480,00 €
Realització de la documentació			
Desenvolupament de la documentació	Analista	70 hores	4.200,00 €
Total		**621 hores**	25.842,50 €

Taula 3.2: Cost total del projecte en funció de l'assignació de tasques als recursos disponibles

Així doncs per a desenvolupar el projecte són necessàries 621 hores de treball i que el conjunt de les tasques tenen un cost associat de 25.842,50 €.

3.6.2 Recursos materials del projecte

Per a desenvolupar el projecte s'ha emprat un portàtil amb un cost de 1.200€ amb el següent software instal·lat:

- Ubuntu 8.04 com a Sistema Operatiu
- Netbeans com a IDE
- DIA com a eina CAD
- Servidor MySQL com a SGBD
- PhpMyAdmin com a Frontend del SGBD

Per tant no hi ha cost relacionat amb el software utilitzat. Caldria considerar, no obstant, el cost associat a la utilització de la màquina vierron, que es troba al laboratori de càlcul del Departament de LSI, malauradament no es disposen d'aquestes dades.

Així doncs el cost associat als recursos materials del projecte és de 1.200€.

3.6.3 Cost total del projecte

Així doncs el cost total del projecte és el següent:

$$Cost\ total = 25.842{,}50\ € + 1.200€ = 27.042{,}50€$$

Observem doncs que es tracta d'un projecte de magnitud mitjana, almenys pel que fa en l'aspecte econòmic.

3.7 Anàlisi de riscos

Durant el transcurs d'un projecte poden succeir alguns esdeveniments que facin perillar la realització del mateix. A continuació s'intentarà detallar quins riscos calia preveure en aquest projecte i la seva resposta.

Risc 1

Risc: Un cop implementada la nova base de dades no obtenim el rendiment necessari

Possibilitats de que succeeixi: Possible

Impacte: Crític

Descripció: Aquest risc es pot materialitzar si no s'ha fet un bon anàlisi de l'arquitectura del sistema i no s'ha dissenyat adequadament la base de dades. En aquest cas caldrà retornar a la fase de disseny tornar a valorar el model conceptual de la base de dades.

Risc 2

Risc: Un cop s'ha dissenyat l'aplicació canvien el format de Genbank

Possibilitats de que succeeixi: Poc probable

Impacte: Poc crític

Descripció: Si canviessin el format de dades amb el que Genbank ofereix la informació caldria readaptar el disseny de l'aplicació de control de dades per a poder tractar aquestes dades.

Risc 3

Risc: El SGBD escollit no és prou potent per implementar el model dissenyat

Possibilitats de que succeeixi: Possible

Impacte: Crític

Descripció: Aquest risc es pot materialitzar si a l'hora d'implementar la base de dades en el SGBD veiem que aquest no és prou versàtil per fer-ho. En aquest cas caldrà canviar de SGBD i, per tant, adaptar el disseny físic de la base de dades.

4 Anàlisi del sistema original

En aquest apartat es farà un anàlisi del sistema original dissenyat i implementat pel doctorand Bernat Gel. Això inclou fer un anàlisi de l'aplicació Genbank per identificar quines necessitats d'informació té. També implica fer proves de rendiment de l'actual base de dades per identificar els colls d'ampolla i analitzar-ne el model lògic.

4.1 Anàlisi del navegador Genexp

Genexp és una aplicació web que permet navegar per la informació genòmica de forma visual d'una manera àgil i còmode. La informació mostrada es troba emmagatzemada en una base de dades a la que s'accedeix mitjançant crides asíncrones al servidor.

Arquitectura de l'aplicació

L'aplicació Genexp està composada per una part servidor i una part client.

Part Client:

La part client de l'aplicació està implementada en Javascript i permet accedir a les diferents funcionalitats de l'eina a través del navegador de l'usuari. Analitzem ara a poc a poc quin és el funcionament de Genexp:

L'aplicació permet escollir en primera instància quin organisme es vol mostrar mitjançant un desplegable com es veu a la figura 4.1. És aquí on es realitza la primera crida a servidor per obtenir els organismes disponibles a la base de dades. Aquesta crida es fa mitjançant la tecnologia AJAX (*Asynchronous JavaScript And XML)* que permet fer comunicació asíncrona entre el navegador i el servidor. Cal notar no obstant que enlloc de XML s'usa el format JSON (*JavaScript Object Notation*) ja que minimitza la sobrecàrrega de dades. Així doncs el script en Javascript llança una crida a servidor per obtenir els organismes disponibles, el servidor genera la resposta en JSON i la retorna al client, aquest la processa i mostra els organismes disponibles en el desplegable, també se'n guarda informació addicional, així com els seus identificadors, per a un ús posterior.

Un cop hem seleccionat un organisme l'aplicació fa una crida a servidor per obtenir els cromosomes que estan disponibles per a aquell organisme. Aquesta crida es fa mitjançant el mateix mètode que l'usat per obtenir els organismes però amb una crida AJAX diferent que usa l'identificador de l'organisme obtingut en el pas anterior. Un cop feta la crida es mostren els cromosomes tal i com es veu a la figura 4.2.

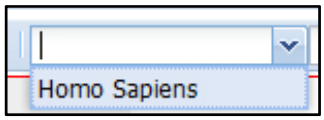


Fig 4.1: Selecció d'organisma

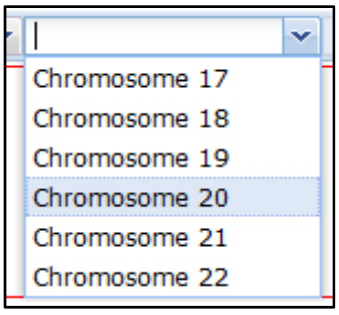


Fig 4.2: Selecció de cromosoma

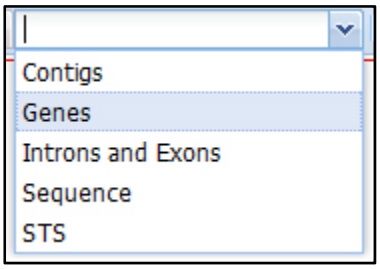


Fig 4.3: Selecció de track

Ja només queda per escollir quin track volem mostrar, és a dir, quin tipus d'informació genètica volem visualitzar per l'organisme. Per tant, caldrà fer una nova crida a servidor per obtenir els tracks disponibles per a aquell organisme (usant el seu identificador per fer la selecció). A continuació l'aplicació mostra els tracks disponibles en un desplegable com veiem a la figura 4.3. A més d'obtenir el nom obtenim informació addicional com el tipus de track del que es tracta (*Feature View* o *Image View)*, més endavant veurem per a que es fa servir aquesta informació. Un cop s'ha seleccionat un track l'aplicació el mostra pel visor tal i com es mostra en la figura 4.4.

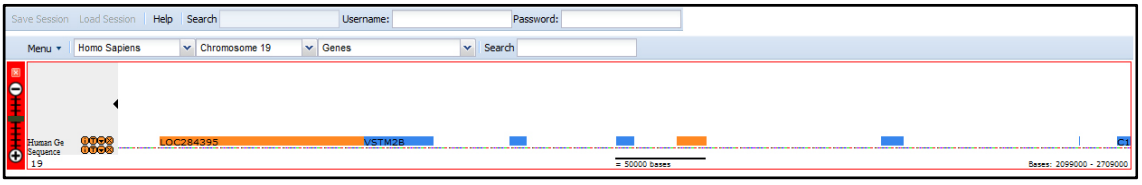


Fig 4.4: Genexp mostrant el track Genes del cromosoma 19 de l'Homo Sapiens

Analitzem ara com ho fa Genexp per mostrar visualment la informació; l'aplicació mostra la informació de dues maneres diferents en funció de la quantitat de característiques a mostrar. Per una banda hi ha l'anomenada *Feature View* utilitzada per a aquells tracks amb menys característiques a mostrar (s'hi mostren fins un màxim de 30.000 característiques). La *Feature View* dibuixa les característiques mitjançant codi HTML, és a dir, són trossos de codi que el navegador web interpreta i mostra per pantalla. És per aquest motiu que hi ha un límit en les característiques que podem arribar a mostrar, ja que si el superéssim el navegador web es quedaria sense memòria disponible. És important notar que quan s'obtenen les característiques d'un track que és del tipus *Feature View* aquestes s'obtenen totes de cop independentment de la zona del cromosoma que s'estigui mostrant. La obtenció d'informació del servidor es fa de la mateixa manera que anteriorment, mitjançant la tecnologia AJAX. A part d'obtenir la posició d'aquestes característiques per a poder-les dibuixar, el client també obté el nom i el tipus de la característica per mostrar la informació per mitjà d'un *ToolTip Text* tal i com es veu a la figura 4.5.

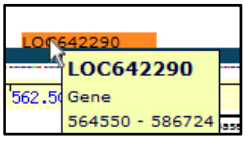


Fig 4.5: ToolTipText

Aquest mètode ens serveix per mostrar aquells tracks amb "poca" informació a mostrar, però com ho fem per mostrar tracks dels que se n'han de mostrar centenars de milers de característiques com els SNPs? Genexp utilitza l'anomenada *Image View* per mostrar aquests tipus de tracks. En aquest cas la part client llança una crida a servidor indicant la zona del cromosoma de la que es volen obtenir dades. El servidor genera una imatge GIF que mostra la informació de forma visual i la guarda en una cache. A continuació envia l'adreça de on es troba aquesta imatge al client. Aquest només ha de mostrar-la pel visor. Per tant, el client només accedeix a zones determinades del cromosoma (que poden ser tot el cromosoma si el zoom és molt ampli).

Ja hem vist com s'ho fa Genexp per mostrar grans quantitats d'informació genòmica de forma visual. Genexp té una funcionalitat que ens permet obtenir informació addicional de cadascuna de les característiques mostrades. Utilitzant els identificadors de les característiques que hem obtingut anteriorment podem fer una crida a servidor per obtenir la informació addicional i mostrar-la en una pantalla tal i com veiem a la figura 4.6.

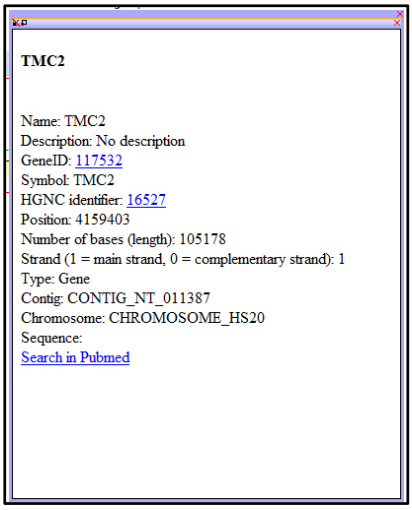


Figura 4.6: Informació addicional del gen TMC2

Així doncs podem afirmar que el client fa quatre tipus de crides:

1. **Crides d'obtenció d'informació bàsica:** Són crides per obtenir quins organismes, cromosomes o tracks hi ha disponibles. Aquestes crides retornaran un conjunt petit d'informació.
2. **Crides d'obtenció d'informació genòmica de tipus Feature View:** Són crides per obtenir característiques que es mostraran pel visor mitjançant codi HTML. La crida retorna tot el conjunt d'informació genòmica determinat per a aquell cromosoma i track. De cada característica mostrada guardem l'identificador, el nom, la posició i el tipus. En aquesta crida, doncs, obtenim un nombre elevat d'informació per part del servidor.
3. **Crides d'obtenció d'informació genòmica de tipus Image View:** Són crides per obtenir característiques que es mostraran pel visor a través d'una imatge. No obstant, el *ToolTip Text* també està operatiu en aquest tipus de visualització i, per tant, caldrà obtenir no només la imatge sinó també l'identificador, el nom, la posició i el tipus de cada característica. Aquestes crides passen com a paràmetre al servidor el cromosoma i el track que s'està mostrant així com el rang de posicions del que volem obtenir les característiques. En aquesta crida també obtenim una quantitat considerable d'informació per part del servidor.

4. **Crides d'obtenció d'informació addicional:** A partir de l'identificador d'una característica fem una crida al servidor per obtenir la informació addicional d'aquella característica. Cal remarcar que aquí només obtenim informació d'una sola característica.

Part Servidor:

Com hem vist, la funció de la part servidor és donar resposta a les crides asíncrones que li fa la part client. Tal i com s'ha explicat anteriorment aquesta informació prové d'una base de dades i el que ha de fer el servidor és connectar-s'hi, extreure'n la informació, i tractar-ne el resultat per generar una resposta JSON, una imatge o ambdues coses. A continuació veurem com la part servidor dóna resposta als quatre tipus de crides que fa el client:

1. **Crides d'obtenció d'informació bàsica:** Aquestes crides són respostes per scripts escrits en PHP. Aquests scripts tenen un funcionament senzill, es connecten a la base de dades, fan una crida per obtenir-ne la informació que calgui i a partir d'aquesta generen la resposta en JSON.
2. **Crides d'obtenció d'informació genòmica de tipus Feature View:** Aquestes crides també són respostes per scripts PHP i es comporten de la mateixa manera que l'anterior, amb la diferència que la quantitat d'informació retornada és molt més elevada. Així doncs a partir de l'identificador del cromosoma i del track, l'script es connecta a la base de dades i n'extreu les característiques corresponents, a continuació construeix la resposta JSON. Cal que les característiques retornades estiguin ordenades creixentment per posició, ja que el client les dibuixarà de forma ordenada.
3. **Crides d'obtenció d'informació genòmica de tipus Image View:** Aquestes crides són ateses per scripts escrits en Perl. La seva tasca consisteix en connectar-se a la base de dades i obtenir la informació genòmica corresponent al cromosoma, track i rang de posicions que s'han passat per paràmetres. A continuació genera una imatge i construeix una resposta JSON amb l'adreça d'aquesta i la informació de les característiques. Cal notar que el sistema utilitza una cache per estalviar-se generar de nou la imatge si aquesta ja ha estat creada anteriorment.
4. **Crides d'obtenció d'informació addicional:** A partir de l'identificador d'una característica un script escrit en PHP es connecta a la base de dades i obté la informació addicional de la característica corresponent. A continuació genera la resposta JSON.

Així doncs veiem que en totes les crides es fa una consulta a la base de dades, cal, per tant, que aquesta doni una resposta ràpida, sobretot pel que fa a les crides que retornen un volum més gran d'informació (tipus 2 i 3). A continuació farem un profund anàlisi de com estava implementada originalment aquesta base de dades.

4.2 Anàlisi de la base de dades original de Genexp

Originalment Genexp disposava d'una base de dades dissenyada i implementada per Bernat Gel. Aquesta base de dades tenia unes deficiències importants en el disseny que provocaven que l'aplicació anés lenta, a més a més s'havia arribat al límit de la seva capacitat de manera que no se li podien carregar més dades. En aquest apartat analitzarem el disseny d'aquesta base de dades per veure quin va ser el punt de partida del projecte.

4.2.1 Model conceptual de la base de dades

Cal remarcar que quan es va iniciar el projecte no es disposava de documentació sobre quin era el model conceptual de la base de dades. Aquest es va haver d'obtenir a partir de l'anàlisi de les relacions implementades en el SGBD.

A la figura 4.7 es mostra el resultat d'aquest procés d'enginyeria inversa.

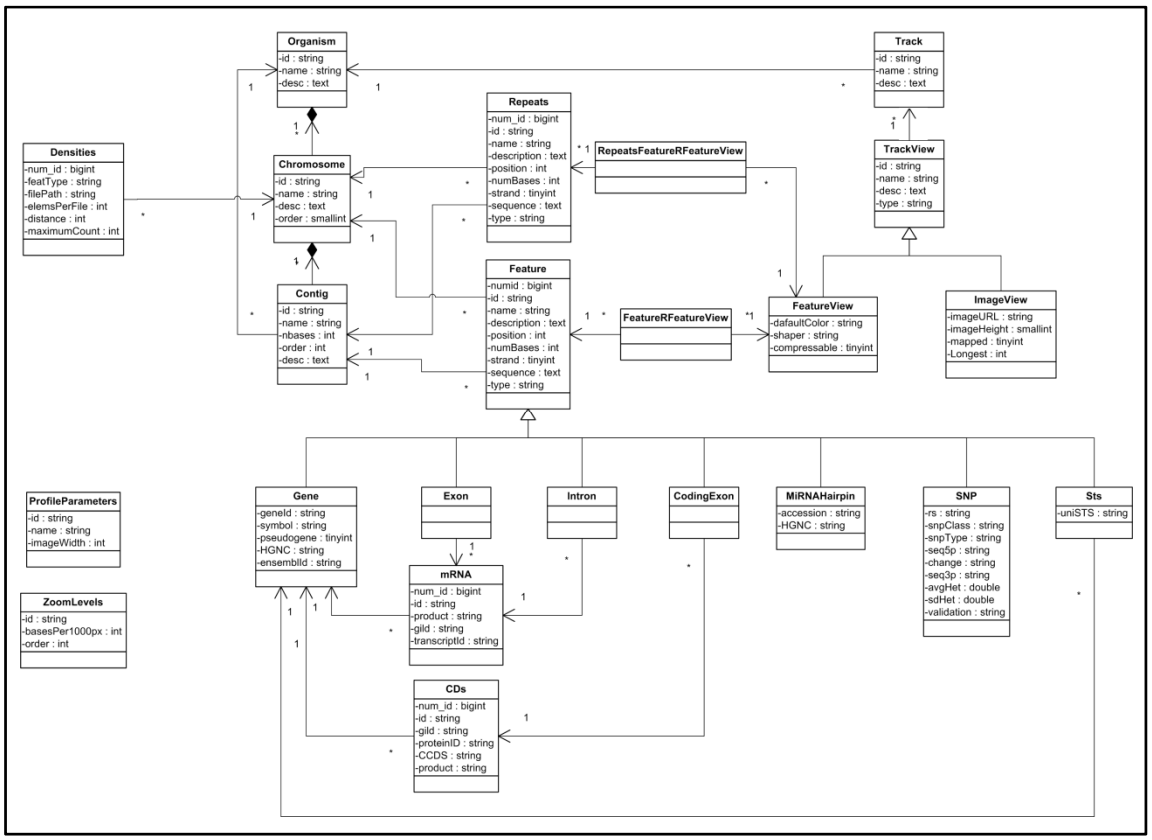


Fig. 4.7: Model conceptual de la base de dades original

A continuació analitzarem les diferents classes i associacions presents en aquest model conceptual:

Primer de tot observem les classes amb informació sobre els organismes, cromosomes i contigs amb els que estan relacionades les característiques i els tracks.

- **Organism:** Conté la informació dels organismes presents en el sistema. Com a atributs tenen un identificador (de tipus String), nom (String) i descripció (de tipus text).

- **Chromosome:** Aquesta classe representa la informació dels cromosomes. Cada un d'aquests cromosomes està relacionat amb un sol organisme i conté com a atributs un identificador (String), nom (String), descripció (text) i ordre del cromosoma dins l'organisme (enter).
- **Contig:** Conté la informació dels contigs presents en els cromosomes. (recordem que els contigs són els diferents conjunts de bases seqüenciades consecutives d'un cromosoma). Els atributs d'aquesta classe són un identificador (String), nom (String), número de bases del contig (enter), ordre del contig en el cromosoma (enter) i descripció (text). Aquests contigs estan relacionats amb el seu corresponent cromosoma i organisme.

A continuació podem observar les classes relacionades amb els tracks:

- **Track:** Representa els diferents tracks de l'aplicació (recordem que un track és cada una de les "files" que es veuen en el visor), aquesta classe conté informació sobre identificador (String), nom (String) i descripció (text). Aquests tracks estan relacionats amb un sol organisme.
- **TrackView:** Aquesta classe pretén donar escalabilitat al sistema. El seu objectiu és que de cada track hi pugui haver diferents "sub-tracks" que mostrin de diferents maneres un mateix track. Per exemple ens podria interessar que el track "gens" es mostrés com una imatge, com codi HTML... la classe conté com a atributs un identificador (String), nom (String), descripció (text) i tipus de TrackView (String), és a dir, el discriminador de la subclasse.
- **ImageView:** Tal i com s'ha vist a l'apartat 4.1 l'ImageView és un mode de visualització de les dades genòmiques que permet representar grans quantitats de dades mitjançant la creació d'imatges. Aquesta classe conté la informació necessària per a representar i generar aquestes imatges. Aquesta informació està composta per la ruta de l'script que genera la imatge (String), l'altura de la imatge i d'altres paràmetres de configuració.
- **Feature View:** El FeatureView és el mode de visualització amb el que el visor mostra la informació genòmica a partir de la generació de codi HTML. Els atributs d'aquesta classe són paràmetres de configuració com el color amb el que s'ha de mostrar la informació pel visor.

És hora d'analitzar les taules relacionades amb el nucli de la base de dades, és a dir, aquelles on es guarda més informació i són més utilitzades:

- **Feature:** Conté la informació bàsica de les característiques (informació que es mostrarà pel visor) presents a la base de dades. En aquesta taula s'hi troba la informació de totes les característiques de tots els cromosomes. Això implica que la quantitat de dades en aquesta taula és molt gran, de fet, en el moment d'analitzar la base de dades (en la qual només hi trobem els 6 cromosomes més petits de l'Homo Sapiens) ens trobem que la taula ocupa uns 2.1GB i té 2,179,947 tuples. En aquesta taula hi trobem atributs que són consultats a l'hora de mostrar la informació bàsica pel visor, però també hi

trobem paràmetres que només són consultats quan es demana la informació addicional corresponent. Val a dir que aquesta taula esdevé la superclasse d'una generalització i que, per tant, només hi trobem les característiques comunes a tots els tipus de característiques. Els seus atributs són un identificador numèric, un identificador nominal, un nom, una descripció, la posició que ocupa dins del contig, el nombre de bases que ocupa, el seu strand, la seva seqüència i el seu tipus, que actua de discriminador en la generalització. A més, manté relacions amb els contigs i cromosomes corresponents.

- **FeatureRFeatureView:** Aquesta taula estableix una relació *-* entre les taules Feature i FeatureView. La idea es que s'ha de permetre que una mateixa característica pugui aparèixer en varis tracks. Una exemplificació d'aquest fet seria que per exemple un mateix gen pogués aparèixer en el track "Gens" i en el track "Gens relacionats amb el càncer" sense que calgués repetir informació del propi gen. Així doncs la taula FeatureRFeatureView només ha de mantenir la relació entre les claus externes de les taules Feature i FeatureView. Aquesta taula té tantes tuples com la taula Feature però en podria tenir més si la relació no fos 1-1 com ho esdevé amb les dades introduïdes.
- **Repeats:** Podríem afirmar que aquesta taula no té massa raó de ser doncs es va crear per a poder fer una presentació de l'aplicació de manera que es poguessin mostrar els Repeats. Té la mateixa estructura que la taula Feature i només es va crear separadament perquè a la taula Feature no hi cabia res més.
- **RepeatsFeatureRFeatureView:** Té el mateix significat que la taula FeatureRFeatureView però amb la clau externa de la taula Repeats.

A continuació observem les diferents subclasses de les característiques que contenen la informació addicional d'aquestes:

- **Gene:** Conté la informació addicional del gen que no es troba a la superclasse. Aquesta consta sobretot dels seus identificadors en les bases de dades genòmiques presents a la xarxa com el anomenat geneId, el símbol, el HGNC o l'ensembleID. També té un atribut que ens indica si es tracta o no d'un pseudogen.
- **mRNA:** No es tracta d'una subclasse de la classe Feature i per tant no esdevé un tipus de característica que es mostri pel visor. La classe mRNA guarda la informació sobre els codis mRNA (DNA missatger). Aquests formen part d'un gen i contenen informació sobre el seu identificador, el producte generat així com d'altres atributs que li corresponen.
- **CDs:** Aquesta classe té les mateixes característiques que la classe mRNA, doncs no és una subclasse de la classe Feature i es tracta d'informació addicional sobre les zones codificants del gen com l'identificador de la proteïna amb la que estan relacionades.

- **Exon i Intron:** Aquestes dues subclasses no tenen atributs propis sinó que es limiten a apuntar al mRNA que els hi correspon.
- **CodingExon:** Aquesta subclasse tampoc té cap atribut sinó que només manté la relació amb la classe CDs vista anteriorment.
- **MiRNAHairpin:** En aquesta subclasse hi trobem la informació corresponent als anomenats MiRNAHairpins dels que es guarden els seus identificadors en les bases de dades genòmiques.
- **SNP:** Aquesta taula conté la informació dels SNPs, és a dir, dels polimorfismes de nucleòtids únics. La informació que es guarda de cada SNP és bastant extensa i sobretot guarda quin és el resultat del polimorfisme. Donada la gran quantitat de polimorfismes existents, aquesta taula té un volum considerable, en el moment d'analitzar les dades inicials veiem que la taula conté 5,851,721 tuples que ocupen 3GB.
- **STS:** Aquesta subclasse conté la informació relacionada amb els STS i manté un apuntador al Gen del que formen part.

Ja només ens queda per veure les taules que contenen informació sobre els paràmetres de configuració del visualitzador:

- **Densities:** Guarda informació necessària a l'hora de dibuixar els SNPs.
- **ProfileParameters:** Guarda aspectes de configuració del sistema.
- **ZoomLevels:** Conté la informació sobre els diferents nivells de zoom presents a l'aplicació, per exemple conté el nombre de bases per cada 1000 pixels visualitzats.

4.2.2 Model lògic de la base de dades

A continuació mostrem el model lògic que surt resulta de la traducció relacional d'aquest model conceptual:

Taules d'organismes, cromosomes i contigs

Organism(id, name, desc)
Chromosome(id, orgId, name, desc, order), {on orgId referencia a Organism}
Contig(id, chrId, orgId, name, nbases, order, desc), {on chrId i orgId referencien a Chromosome}

Taules de tracks

Track(id, name, desc, orgId), {on orgId referencia a Organism}
TrackView(id, trackId, name, desc, type), {on trackId referencia a Track}
FeatureView(trackViewId, trackId, defaultColor, shaper, compressable), {on trackViewId referencia a TrackView}
ImageView(trackViewId, trackId, imageURL, imageHeight, mapped, longest), {on trackViewId referencia a TrackView}

Taules de features

Feature (num_id, id, name, position, numBases, strand, type, contigId, chrId, orgId, description, sequence), {on contigId, chrId i orgId referencien a Contig}
FeatureRFeatureView (FeatureViewId, FeatureId) { on FeatureViewId referencia a FeatureView i FeatureId referencia a Feature }

Taules de Repeats

Repeat (num_id, id, name, position, numBases, strand, type, contigId, chrId, orgId, description, sequence), {on contigId, chrId i orgId referencien a Contig }
RepeatRFeatureView (FeatureViewId, RepeatId) { on FeatureViewId referencia a FeatureView i RepeatId referencia a Repeat }

Taules de les subclasses de feature

Gene(featureId, geneId, symbol, pseudogene, HGNC, ensemblId, desc, sequence), {on featureId referencia a Feature}
Exon(featureId, desc, sequence), {on featureId referencia a Feature}
ExonRmRNA(exonID, mrnaId), {on exonID referencia a Exon i mrnaId referencia a mRNA)
mRNA(numId, id, product, gild, transcriptID, geneFeatureId), {on geneFeatureId referencien a Gene)
CDs(numId, id, gild, proteinId, CCDS, product, geneFeatureId), {on geneFeatureId referencien a Gene)
Intron(featureId, desc, sequence, mrnaId), {on featureId referencia a Feature i mrnaID referencia a mRNA}
CodingExon(featureId, desc, sequence, cdsId), {on featureId referencia a Feature i cdsId referencia a CDs}
miRNAHairpin(featureId, accession, HGNC, desc, sequence), {on featureId referencia a Feature}
SNP(featureId, rs, snpClass, snpType, seq5p, change, seq3p, avgHet, sdHet, validation, desc, sequence), {on featureId referencia a Feature}
Sts(featureId, uniSTS, desc, sequence, geneOrgId, geneChrId, geneFeatureId), {on featureId referencia a Featurei i geneFeatureId referencien a Gene}
MatureMicroRNA(featureId, accession, desc, sequence, miRNAHairpinFeatureId), {on featureId referencia a Feature i miRNAHairpinFeatureId referencia a miRNAHairpin}

Taules de configuració

Densities(num_id, fearType, filePath, elemsPerFile, distance, maximumCount, chrId, orgId), {on chrId i orgId referencien a Chromosome}
ProfileParamaters(id, name, imageWidth)
ZoomLevels(id, basesPer1000px, order)

4.2.3 Model físic de la base de dades

La base de dades es troba implementada en un servidor MySQL a la màquina "vierron" del departament de Llenguatges i Sistemes Informàtics de la UPC. Aquesta màquina té les especificacions següents:

Model	Fire 280R
Fabricant	Sun Microsystems
CPU	2 CPUS UltraSPARC III a 750MHz
Memòria	4 GB de RAM
Disc	2 discos FC-AL SCSI de 10.000 RPM y 32 Gb

Taula 4.1: Especificacions tècniques de la màquina Vierron

MySQL pot usar diferents motors a l'hora d'implementar la base de dades, en el cas de la base de dades de Genexp original s'utilitza el motor MyISAM. Aquest motor es caracteritza per la seva simplicitat però també per ser capaç de donar resposta ràpida a crides simples. És degut a aquesta simplicitat que les seves opcions són molt limitades, per exemple no implementa claus externes ni permet gestionar transaccions. És per això que a la base de dades original de Genexp no s'assegura integritat referencial entre les relacions. Aquest fet implica que si no es fa un bon control a l'hora d'introduir les dades ens podem trobar que hi hagi dades incorrectes o incoherents.

Pel que fa a les estructures d'accés (índexos) implementades a la base de dades per tal d'intentar millorar els temps d'accés podem observar que n'hi ha un bon nombre i que algunes d'elles són innecessàries doncs no són necessàries per les crides que es realitzen. Per exemple veiem que a la taula Feature hi trobem cinc estructures d'accés diferents:

Nom columna
num_id
id
Chromosome_id
type
position_index

Taula 4.2: Índexs de la taula Feature

Com veurem en el següent apartat els camps num_id i type no són usats en les crides i per tant no és necessari crear una estructura d'accés que els indexi. Aquesta redundància es dóna en moltes de les taules, serà necessari per tant decidir quines estructures d'accés són necessàries per tal de minimitzar els temps d'accés i descartar aquelles que no tenen cap utilitat.

4.3 Identificació de les crides efectuades

Ara que ja hem vist quin és el funcionament de l'explorador i hem analitzat el model lògic de la base de dades podem veure quines són les crides SQL efectuades per l'aplicació a la base de dades. La tipologia d'aquestes crides es correspon a la utilitzada entre el client i el servidor del navegador que hem vist en el punt 4.1 de la memòria. Així doncs analitzarem les crides SQL en cada un d'aquests tipus de crides:

1. **Crides d'obtenció d'informació bàsica:** Són crides per obtenir quins organismes, cromosomes o tracks hi ha disponibles a la base de dades. La crida per obtenir els organismes seria la següent:

```
SELECT `id`, `name`
FROM `Organism`
ORDER BY `name`
```

   També podem identificar la crida que obté els cromosomes d'un organisme donat:

```
SELECT `id`, `name`
FROM Chromosome
WHERE `OrganismID` = 'ORGANISM_human'
ORDER BY `order`
```

2. **Crides d'obtenció d'informació genòmica de tipus Feature View:** Són crides per obtenir característiques que es mostraran pel visor mitjançant codi HTML. Podem observar la crida que ens retorna els gens del cromosoma 18:

```
SELECT Feature.id, Feature.name,
       Feature.description, Feature.position,
       Feature.numBases, Feature.strand, Feature.type,
       Feature.ContigID,Feature.ChromosomeID
FROM Feature, FeatureRFeatureView
WHERE Feature.id = FeatureRFeatureView.FeatureID
      AND FeatureRFeatureView.FeatureViewID =
      'TRACKVIEW_human_genes'
      AND Feature.ChromosomeID = 'CHROMOSOME_HS18'
ORDER BY Feature.position;
```

   En aquesta crida podem veure com és necessari fer un 'join' entre la taula Feature i la taula FeatureRFeatureView per a poder seleccionar les característiques del track desitjat.

3. **Crides d'obtenció d'informació genòmica de tipus Image View:** Són crides per obtenir característiques que es mostraran pel visor a través d'una imatge. A continuació podem observar la crida que ens retorna els SNPs:

```
SELECT Feature.id, Feature.name,
       Feature.position, Feature.numBases,
       Feature.type FROM Feature, FeatureRFeatureView
WHERE Feature.id = FeatureRFeatureView.FeatureID
      AND FeatureRFeatureView.FeatureViewID =
      'TRACKVIEW_human_SNP'
      AND Feature.ChromosomeID = 'CHROMOSOME_HS21'
      AND Feature.position > 1512000
      AND Feature.position < 41512000
ORDER BY Feature.position;
```

   En aquesta crida podem veure com introduïm un rang específic de posicions sobre el que fer la cerca de les característiques.

4. **Crides d'obtenció d'informació addicional:** A partir de l'identificador d'una característica fem una crida al servidor per obtenir la informació addicional d'aquella característica. Aquí podem veure la crida per obtenir informació addicional d'un gen determinat:

```
SELECT * FROM Gene
WHERE FeatureID = 'FEATURE_8'
```

4.4 Anàlisi dels temps de resposta

En aquest punt analitzarem els resultats obtinguts a partir dels tests de rendiment que es van fer sobre la base de dades original. Els objectius d'aquestes proves són per una banda obtenir de forma quantitativa valors que ens permetin comparar el rendiment de la base de dades original amb el rendiment de la base de dades optimitzada i, per una altra, intentar identificar colls d'ampolla i mancances de disseny que repercuteixen en els temps de resposta.

L'anàlisi fa més èmfasi en les crides de tipus 2 i 3 (veure punt anterior), que són les que retornen les diferents característiques dels cromosomes mostrats i les que retornen més informació, també són les que tarden més temps. A continuació analitzarem quines són les variables en aquestes crides.

Identificació de les variables de les crides

Primer de tot recordem en què es diferencien les crides de tipus 2 i 3, mentre que les de tipus 2 (obtenen informació genòmica de tipus Feature View) obtenen informació de tot el cromosoma, en les crides de tipus 3 (obtenen informació genòmica de tipus Image View) cal especificar un rang de posicions determinat.

Per tant, els paràmetres d'aquestes dues crides són:

Feature View	Image View
FeatureViewID	FeatureViewID
ChromosomeID	ChromosomeID
	posInicial
	posFinal

Taula 4.3: Paràmetres de les crides

Per a realitzar els tests s'han programat petites aplicacions en Java que han estat executades a la mateixa màquina on hi ha el SGBD.

A continuació es detallaran els detalls dels tests realitzats i es mostraran els resultats obtinguts.

Test 1: Obtenció de característiques de tipus Feature View

L'objectiu d'aquest test és veure com es comporta el sistema a l'hora de retornar característiques d'un track de tipus Feature View. Aquestes crides retornen un nombre de tuples mitjà ja que de ser molt elevat s'hauria d'usar el dibuixat en imatge i usaríem una Image View.

En aquest test farem varies proves en funció de diferents FeatureViewID; realitzarem crides en els tracks de contigs, gens i STS mentre que deixarem fix el cromosoma.

Track dels contigs:

S'han fet 100 repeticions de la crida que ens retorna els contigs d'un cromosoma determinat, en aquest cas del 18. Donat que en un cromosoma hi ha molts pocs contigs aquesta és una crida que retorna poques tuples. Els resultats obtinguts són els següents:

Variable estadística	Valor
Mitjana del temps de resposta	0,14 s
Desviació estàndard del temps de resposta	0,02 s

Taula 4.4: Resultats del test 1, track dels contigs

Track dels gens:

En aquest cas s'han fet 100 crides per obtenir els gens del cromosoma 18. En un cromosoma com aquest hi trobem uns 600 gens, o sigui que el nombre de tuples retornades tampoc és molt elevat.

Variable estadística	Valor
Mitjana del temps de resposta	0,72 s
Desviació estàndard del temps de resposta	0,1 s

Taula 4.5: Resultats del test 1, track dels gens

Track dels STS:

Finalment hem fet 100 repeticions de la crida que retorna els sts del cromosoma 18. Aquí el nombre de característiques retornades és més elevat ja que en el cromosoma 18 hi trobem 8308 STS diferents.

Variable estadística	Valor
Mitjana del temps de resposta	7,28
Desviació estàndard del temps de resposta	0,14

Taula 4.5: Resultats del test 1, track dels STS

Podem observar que els temps de resposta d'aquestes crides no són massa elevats. De fet en el cas dels contigs i dels gens, aquest temps és més que acceptable. No obstant en el cas dels STS cal millorar-ne el rendiment.

Test 2: Obtenció de característiques de tipus Image View

L'objectiu d'aquest test és veure com es comporta el sistema a l'hora de retornar característiques d'un track de tipus Image View. Aquestes crides es caracteritzen per retornar un gran nombre de tuples i que, degut a la uniformitat en el posicionament de les característiques al llarg dels cromosomes, el nombre de tuples retornat és proporcional a la longitud del rang especificat. Així doncs s'ha deixat fix el track (en aquest cas s'ha elegit el track dels SNPs) i el cromosoma (s'ha elegit el cromosoma 18 del Homo Sapiens) i s'ha anat variant aleatòriament el rang de posicions.

Després de fer 1000 repeticions d'aquesta prova hem obtingut els següents resultats:

Variable estadística	Valor
Mitjana del temps de resposta	23,38 s
Desviació estàndard del temps de resposta	11,92 s
Coeficient de correlació entre el nombre de tuples retornades i el temps de resposta	0,96
Coeficient de correlació entre la longitud del rang de posicions i el temps de resposta	0,95

Taula 4.6: Resultats del test 2

Amb els resultats del test s'han elaborat els següents gràfics:

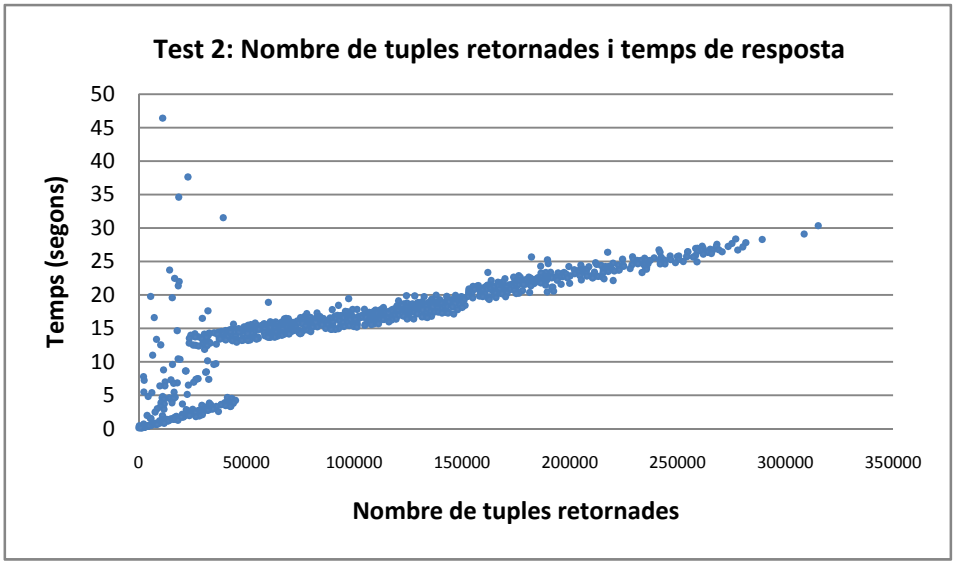


Fig 4.8: Resultat del test 2 comparant el nombre de tuples retornades amb els temps de resposta obtinguts

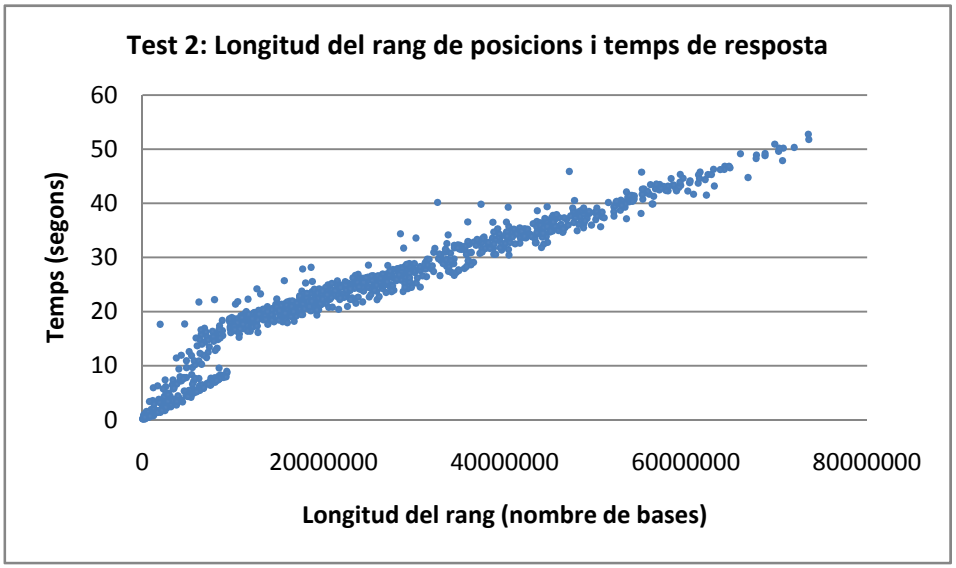


Fig 4.9: Resultat del test 2 comparant la longitud del rang de posicions amb els temps de resposta obtinguts

A la figura 4.8 podem veure com la base de dades té comportaments diferents en funció del nombre de tuples retornades. Es pot apreciar com varia la recta en funció de si el nombre de tuples a retornar és superior o inferior a un valor que oscil·la sobre les 50.000 tuples.

Amb aquestes dades podem concloure clarament que com més tuples s'han de retornar més es triga en fer-ho. Arribats a aquest punt ens hem de fer la següent pregunta:

> Els temps de resposta són conseqüència directa i inevitable de l'elevat nombre de dades que s'ha de retornar?

Si això fos cert poc es podria fer per millorar el rendiment de la base de dades doncs els temps de resposta serien alts sempre que el nombre de dades a retornar fos alt.

No obstant, podem observar un fet que influeix de forma determinant en aquesta clara proporcionalitat. Si observem la crida mostrada a continuació hi observem clarament una 'Join' entre les taules Feature i FeatureRFeatureView. Aquest 'Join' és necessari per a poder destriar quines Features són del track que ens interessa.

```
SELECT Feature.id, Feature.name,
       Feature.position, Feature.numBases,
       Feature.type FROM Feature, FeatureRFeatureView
WHERE Feature.id = FeatureRFeatureView.FeatureID
      AND FeatureRFeatureView.FeatureViewID =
      'TRACKVIEW_human_SNP'
      AND Feature.ChromosomeID = 'CHROMOSOME_HS21'
      AND Feature.position > 1512000
      AND Feature.position < 41512000
ORDER BY Feature.position;
```

La conseqüència d'això és que per cada tupla de la taula FeatureRFeatureView del track desitjat (en aquest cas del track SNP) cal buscar a la taula Feature si existeix la tupla que li correspon i si aquesta compleix la resta de restriccions (cromosoma i posició). Això provoca que els temps es disparin si el nombre de tuples a les taules és molt gran i, que a més, els temps de resposta siguin proporcionals al nombre de tuples retornades.

En el següent test intentarem demostrar que si evitem el 'Join' els temps de resposta disminueixen notablement.

Test 3: Obtenció de característiques evitant el 'Join'

L'objectiu d'aquest test és demostrar que el 'Join' que es realitza a la crida repercuteix negativament en el rendiment de la base de dades. Abans de poder prescindir del 'Join' cal que trobem un mètode alternatiu per a obtenir les mateixes dades sense el 'Join'. Aquesta informació la podem treure del camp type de la classe Feature. Aquest camp està relacionat amb el track que volem, en alguns casos aquesta relació és 1 a 1, com en el cas dels SNPs on el track corresponent només està relacionat amb les instàncies de la classe Feature que són del tipus SNP. Altres vegades ens trobem que un mateix track

pot estar relacionat amb varis tipus, aquest és el cas del track dels Introns i Exons, que tot i ser un sol track està relacionat amb dos tipus diferents (Introns i Exons).

En aquest test farem una selecció sobre la taula Feature per tal d'obtenir totes les característiques que són del tipus SNP.

La crida realitzada en aquest test és la següent:

```
SELECT Feature.id, Feature.name, Feature.position,
       Feature.numBases, Feature.type
FROM Feature
WHERE Feature.type='SNP'
      AND Feature.ChromosomeID = 'CHROMOSOME_HS18'
      AND Feature.position > $POSINI
      AND Feature.position < $POSFI
ORDER BY Feature.position;
```

S'han realitzat 1000 repeticions d'aquesta crida variant aleatòriament el rang de posicions. Després de realitzar el test s'han obtingut els següents resultats:

Variable estadística	Valor
Mitjana del temps de resposta	13,9 s
Desviació estàndard del temps de resposta	6,89 s
Coeficient de correlació entre el nombre de tuples retornades i el temps de resposta	0,84
Coeficient de correlació entre la longitud del rang de posicions i el temps de resposta	0,83

Taula 4.7: Resultats del test 3

En el següent gràfic mostrem la relació entre les tuples retornades i els temps de resposta. En el gràfic també s'hi mostren els resultats del test anterior per poder comparar.

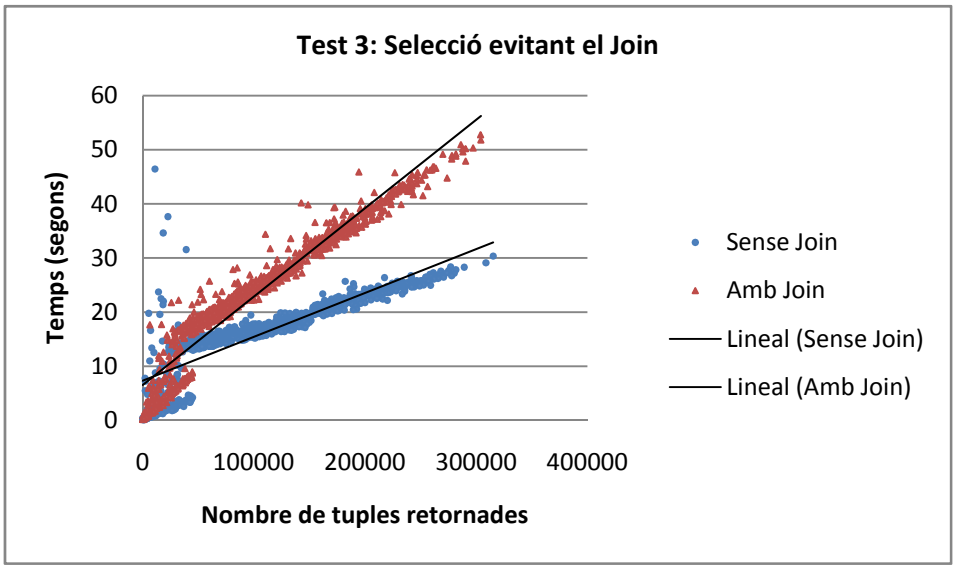


Fig 4.10: Resultat del test 3 comparant el nombre de tuples retornades amb els temps de resposta obtinguts

Podem veure com si evitem el Join obtenim un guany mitjà d'uns 10 segons en el temps de resposta tot i que, com veiem en el gràfic, aquest guany és molt més pronunciat quan el nombre de tuples retornades és més elevat. Així doncs podem concloure que evitar el join és una manera efectiva de minimitzar els temps de resposta.

Test 4: Obtenció d'informació bàsica

En aquest test es vol comprovar quin és el temps de resposta per obtenir informació bàsica com, per exemple, obtenir els organismes disponibles a la base de dades o obtenir els cromosomes relacionats amb aquets cromosomes.

S'han fet 1000 repeticions de la crida que ens retorna els organismes disponibles a la base de dades i s'han obtingut els resultats següents:

Variable estadística	Valor
Mitjana del temps de resposta	0,04 s
Desviació estàndard del temps de resposta	0,05 s

Taula 4.7: Resultats del test 4, obtenció d'organismes

També s'han fet 1000 repeticions de la crida que ens retorna els cromosomes d'un organisme determinat i s'han obtingut els resultats següents:

Variable estadística	Valor
Mitjana del temps de resposta	0,04 s
Desviació estàndard del temps de resposta	0,06 s

Taula 4.8: Resultats del test 4, obtenció de cromosomes

Amb els resultats a la ma podem concloure que aquests són més que acceptables i que poc podrem fer per millorar-los.

Test 5: Obtenció d'informació addicional

L'objectiu d'aquest test és avaluar el rendiment del sistema a l'hora d'obtenir la informació addicional (de les subclasses) d'una característica determinada.

Per fer-ho s'han fet 1000 crides variant la característica de la que es vol aquesta informació addicional de tipus SNP. S'han obtingut els següents resultats:

Variable estadística	Valor
Mitjana del temps de resposta	0,12s
Desviació estàndard del temps de resposta	0,05 s

Taula 4.9: Resultats del test 5, obtenció d'informació addicional

Podem concloure que aquesta crida tampoc suposa un problema de rendiment donat que els temps de resposta són més que acceptables. Tot i que es tracta d'una taula molt gran (5,851,721 tuples i 3,102 GB), gràcies a les estructures d'accés obtenim un accés molt ràpid quan volem accedir a una sola tupla a partir de la seva clau primària.

4.5 Identificació dels principals problemes

Un cop realitzats els tests hem pogut veure que els temps d'accés són massa alts quan hem d'obtenir moltes tuples, aquest fet fa que l'aplicació no sigui còmode d'utilitzar. Després de l'anàlisi del comportament de l'aplicació, dels esquemes lògic i físic i d'haver realitzat els diferents tests podem ja identificar alguns aspectes millorables del disseny de la base de dades, aquests aspectes hauran de ser considerats a l'hora de redissenyar el model:

- **Identificadors:** En la majoria de classes els identificadors no són numèrics i autoincrementables sinó que es tracta de strings generats per l'aplicació que introdueix les dades. Aquest fet fa que les comparacions efectuades per l'SGBD siguin molt més lentes.
- **Mala distribució dels atributs:** Tal i com hem vist en la descripció de l'aplicació hi ha dos tipus d'accés a la informació genòmica, per una banda l'accés que ens dona la informació necessària per "pintar" les característiques pel visor i per l'altra l'accés que ens permet obtenir informació addicional d'una característica concreta. Aquesta informació hauria d'estar repartida de manera que s'optimitzessin aquests accessos, per tant podem concloure que hi ha camps a la superclasse Feature que podrien estar a les subclasses.
- **Join:** És conegut que el "Join" és la operació més costosa de qualsevol SGBD, per tant és una operació que caldria evitar. Tal i com hem vist en el test 3 obtenim una millora de rendiment considerable si així ho fem.
- **Atributs i classes innecessaris:** Cal replantejar-se la necessitat de guardar certs atributs i classes a la base de dades doncs potser no es faran servir mai pel navegador o simplement no són necessaris, cal doncs, parlar amb el desenvolupador del navegador, el doctorand Bernat Gel, per tal d'obtenir el model de dades definitiu.
- **Índexs innecessaris:** Hi ha estructures d'accés creades en el SGBD que no són necessaris donades les crides que es realitzen i que només esdevenen una despesa d'espai.
- **Tipus de dades dels atributs:** Hi ha atributs que no estan representats per un tipus de dades de forma òptima, cal definir quin rang de valors utilitza cada atribut i assignar el tipus de dades que millor s'hi ajusti per tal de minimitzar l'espai utilitzat.
- **Classes mal situades en el model:** Hi ha mancances en el disseny, com per exemple el cas de la classe Repeat, la qual s'hauria de tractar com una subclasse de Feature més.

Veiem doncs que la base de dades té molts aspectes millorables i que poden repercutir de forma clara en una millora del rendiment. Així doncs cal definir un model de dades que s'adeqüi de forma òptima a les necessitats de l'aplicació. Aquesta tasca de disseny la descriurem en el següent apartat.

5 Optimització de la base de dades

Es aquest punt es descriurà el treball realitzat pel que fa a l'optimització de la base de dades així com els resultats obtinguts.

5.1 Disseny de la base de dades

Tal i com hem descrit en la introducció i en els capítols anteriors l'objectiu fonamental d'aquest projecte és millorar el rendiment de la base de dades de Genexp i augmentar-ne la seva escalabilitat. L'assoliment d'aquest objectiu passa necessàriament per un redisseny lògic i físic de la base de dades. En aquest apartat descriurem quines han estat les decisions preses pel que fa a aquest redisseny.

5.1.1 Model conceptual de la base de dades

Després de debatre quin havia de ser el model de dades amb el desenvolupador de Genexp s'ha arribat a un model de dades definitiu. Aquest respon a certes consideracions generals preses durant el procés d'elaboració i que descric a continuació:

- La base de dades de genexp no ha de ser una base de dades genòmica, sinó una base de dades que doni resposta a les necessitats del navegador. Això implica que només cal guardar aquella informació que es mostri pel visor.
- Ha de respondre a criteris de millora de rendiment més que a les pautes tradicionals del disseny de models conceptuals. Això significa que en alguns casos el disseny no s'ajusta a les directives apreses a les assignatures d'especificació de software, sinó que respon l'objectiu de minimitzar els temps de resposta.
- Ha de prevaldre la simplicitat. Un esquema simple és més senzill de mantenir, però també el pot fer més escalable i eficaç.

Tal i com hem vist s'ha fet un treball de redisseny del model conceptual de dades que ha permès obtenir uns temps de resposta menors. El punt de partida ha estat el model conceptual original i amb les mancances detectades i les línies de treball descrites anteriorment s'ha elaborat un nou model de dades.

A la figura 5.1 hi podem veure el model de dades definitiu. La principal diferència que podem veure amb el model original (figura 4.7) és el guany en simplicitat, hi ha menys classes i per tant també menys relacions. També podem veure com algunes classes han canviat de lloc en el model i com en altres casos només han estat els atributs els que han canviat de classe. És hora de descriure en detall quins han estat aquests canvis i perquè han esdevingut d'aquesta manera.

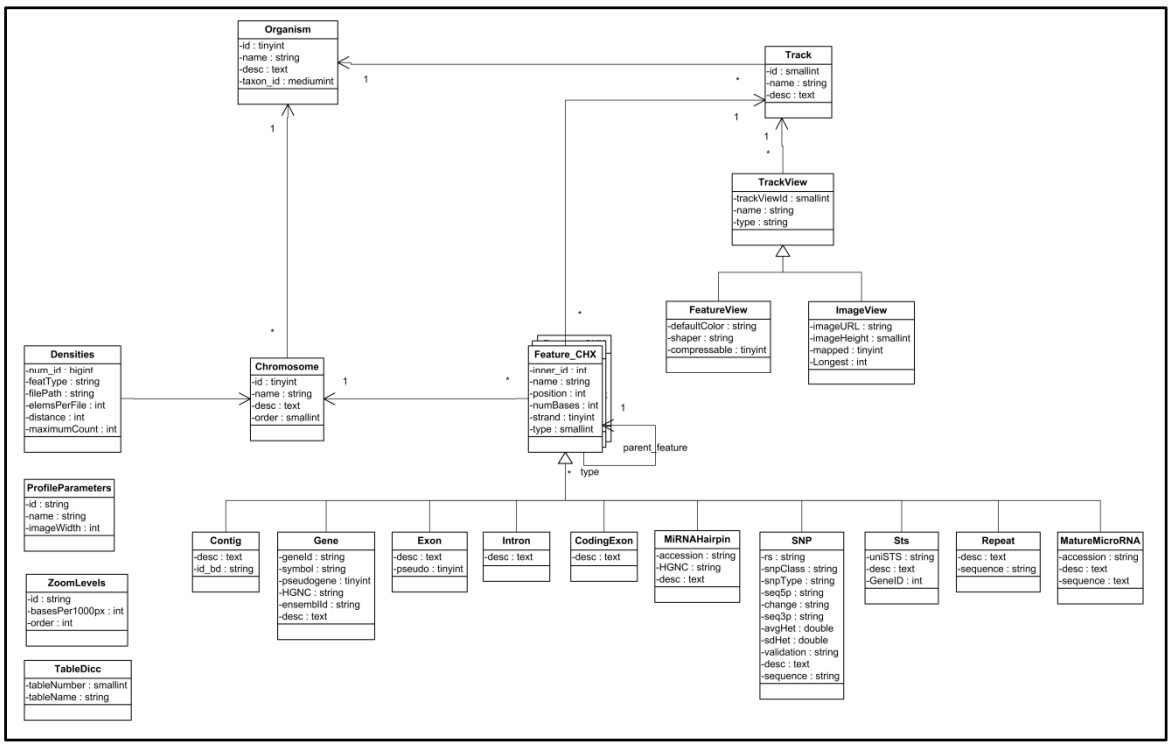


Fig 5.1: Model conceptual resultant

A més de guanyar en simplicitat, amb aquest model de dades també guanyem en escalabilitat perquè, amb els canvis que veurem a continuació, les dades estan més ben distribuïdes i el model ha de canviar molt poc en el cas de que vulguem poder tractar un nou tipus de dades genòmiques, de fet, pràcticament només caldria afegir una subclasse de Feature per a poder-ne guardar la informació addicional.

Moviment de la classe Contig

En el diagrama original, la classe contig, es trobava relacionada amb la classe organisme i cromosoma mitjançant una agregació. Donat que s'ha considerat que els contigs només són un tipus de característica més, com ho poden ser els gens, s'ha decidit que els contigs passin a ser una subclasse de Feature. Veiem aquest canvi a la següent figura.

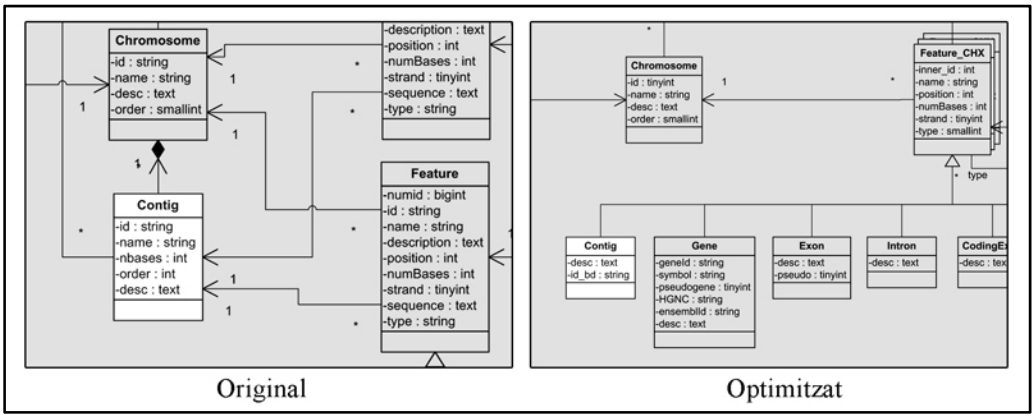


Fig. 5.2: Moviment de la classe contig

Restructuració dels repeats

Si observem el diagrama original, els repeats tenien una entitat alternativa a la classe Feature però es comportaven de la mateixa manera i tot tenia la mateixa estructura. Això es devia a que es va necessitar poder guardar repeats i a la taula Feature no hi cabia res més. Així doncs s'ha decidit donar als Repeats el mateix tractament que a les Features, i , per tant, passen a ser subclassses de la classe Feature. Així doncs també eliminem la classe RepeatsFeatureRFeatureView que lligava la classe repeats amb la classe FeatureView. Ho veiem a la següent figura.

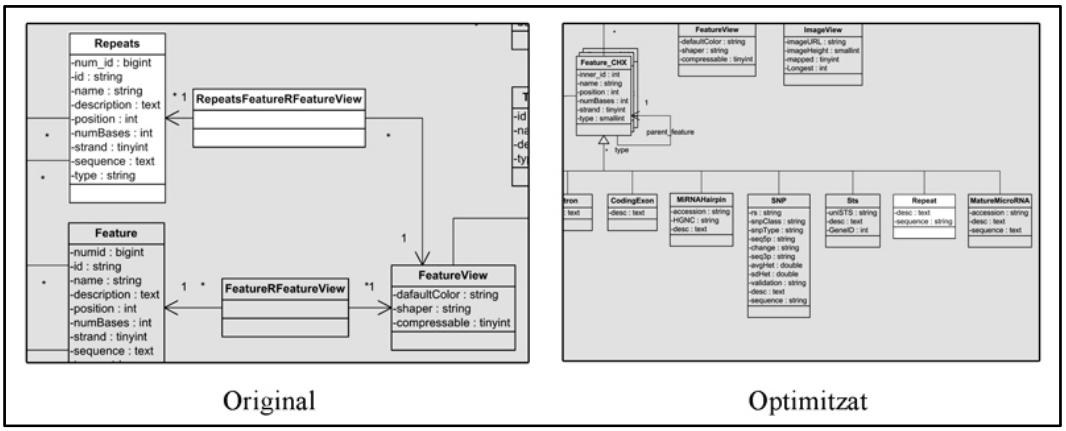


Fig. 5.3: Moviment de la classe repeats

Eliminació de classes d'informació addicional

Una de les línies de treball que s'han definit en el apartat anterior consistia en considerar que la base de dades no ha de ser una base de dades genòmica sinó que només ha de contenir la informació requerida per Genexp. En el model original hi guardavem molta informació que mai s'arribava a mostrar pel navegador, s'ha decidit eliminar aquesta informació, per tant, totes les relacions i classes relacionals que hi havia entre les subclasses han sigut eliminades. Veiem aquest canvi a continuació.

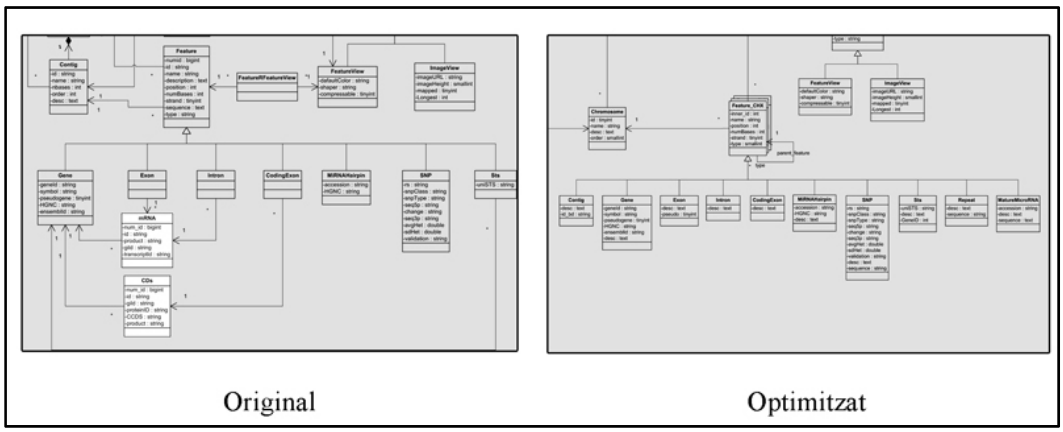


Fig. 5.4: Eliminació de les classes d'informació addicional

Particionament de la classe Feature

Tal i com hem pogut veure, l'accés a la classe Feature esdevé el punt clau del rendiment d'aquesta base de dades. Fins ara, totes les dades de tots els cromosomes estaven a la mateixa taula tot i que l'accés només es fa per un sol cromosoma i no hi ha relacions entre característiques de diferents cromosomes. Per tal de millorar el rendiment de la base de dades és important que les taules de les que es vol fer un accés ràpid fossin el més petites possible, d'aquesta manera permetem que hi càpiguen en memòria se'n podrà fer un accés molt més ràpid per a les properes consultes. Per tal de reduir l'espai podem fer dues coses, reduir la mida de cada tupla i reduir el nombre de tuples. Analitzem de moment aquesta darrera possibilitat. Si hi ha independència referencial entre cromosomes, podem fàcilment assignar una taula diferent a cada cromosoma. D'aquesta manera aconseguirem que tots els accessos consecutius a un cromosoma es facin sobre una única taula que no contindrà la informació d'altres cromosomes, que de fet, no necessitem. Així doncs, el que s'ha fet és crear una taula per cada cromosoma i que té per nom "Feature_chX" on X és l'identificador del cromosoma. El control de quines taules hi ha creades es deixa per l'aplicació de control.

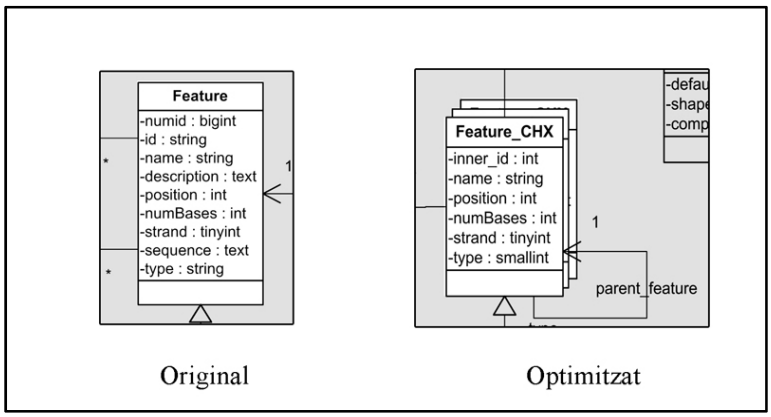


Fig. 5.5: Partició de la classe Feature

Redistribució dels atributs de la classe Feature

Si en el canvi anterior l'objectiu consistia en reduir el nombre de tuples, l'objectiu d'aquest canvi és disminuir la mida de cada una de les tuples. Per tal de fer això cal, per una banda, determinar quina informació cal que estigui a la superclasse i quina no, i, per una altra banda, ajustar els tipus de dades a les necessitats reals de cada un dels atributs. Si ens centrem, per ara, en la primera via podem veure que en el model original hi trobàvem camps com la descripció que, si bé el camp és comú a totes les subclasses i per tant, seria lògic que anés a la superclasse, veiem que es tracta d'una informació que el visor no requereix fins que ha d'extreure informació addicional de la subclasse, així doncs podem concloure que és lògic que aquest atribut es guardi a la subclasse enlloc de a la superclasse.

Canvi de tipus dels identificadors

A la versió original, la majoria d'identificadors (claus primàries) eren de tipus string. Aquest fet repercuteix negativament en el rendiment de les consultes, ja que les

comparacions entre strings a l'hora de retornar les tuples van molt més lentes que les comparacions entre valors numèrics. Per tant, s'han canviat els identificadors per a que només estiguessin implementats amb valors numèrics.

Ajustament dels tipus de dades

Per tal de reduir la mida de les taules és important reduir al màxim la mida de cadascun dels atributs, cal doncs, seleccionar els tipus de dades que millor s'ajustin al rang de valors de cada atribut. De moment ens quedem només amb la idea i deixem pel disseny físic de la base de dades la descripció de quins són els tipus de dades definitius.

Eliminació de la classe FeatureRFeatureView

Recordem que una de les principals mancances de la base de dades original era que esdevenia necessari fer un "Join" entre la taula Feature i la taula FeatureRFeatureView per tal d'obtenir les Features que es corresponien amb un Track específic. Com hem dit, el "Join" és la operació més costosa de qualsevol SGBD, per tant, era interessant intentar evitar-la. Cal considerar que la relació entre Tracks i Features és una relació *-*, això implica que no podem eliminar la classe relacional fàcilment, ja que es fa necessària per poder fer la selecció. Així doncs ha calgut buscar una solució alternativa que descrivim a continuació.

Com hem vist cal permetre que un mateix Feature pugui pertànyer a varis Tracks, sense usar una taula addicional només ens queda la opció de repetir tuples a la taula Feature en les que només es diferenciarà el Track. Veiem ara de forma gràfica instàncies amb els dos models:

Model Original

Classe Feature
Feature Id
1
2
3
4

Classe FeatureRFetureView	
Feature Id	TrackId
1	1
1	2
2	1
3	2
4	1

Classe FeatureView	
FeatureViewId	Nom
1	Gens
2	Gens relacionats amb el càncer

Taula. 5.1: Relació de features amb el track (model original)

Model Optimitzat

Classe Feature	
Feature Id	TrackId
1	1
2	2
3	1
4	2

Classe FeatureView	
TrackId	Nom
1	Gens
2	Gens relacionats amb el càncer

Taula. 5.2: Relació de features amb el track (model optimizat)

Les tuples 1 i 2 de la classe Feature contenen la mateixa informació a la resta de camps, però es diferencien en el trackId i en l'identificador.

Aquest canvi introdueix un problema d'integritat referencial: Què passa si esborro una de les tuples que comparteixen informació? La resposta és clara, cal esborrar les seves tuples relacionades ja que totes elles formen una sola entitat. Per a poder fer-ho hem afegit un nou camp a la taula Feature que ens relaciona les tuples que formen part d'una sola entitat. Aquest fet ens porta directament cap a l'establiment d'una jerarquia entre les tuples d'una mateixa entitat, on hi ha una tupla arrel i unes tuples filles, quan esborri la tupla arrel he de poder identificar les tuples filles que li corresponen i esborrar-les. La idea és que aquest camp contingui un '0' a l'arrel de la jerarquia i l'identificador de l'arrel a les tuples filles. Veiem-ho de forma gràfica:

Classe Feature		
Feature Id	**TrackId**	**ParentId**
1	1	0
2	2	1
3	1	0
4	1	0

Classe FeatureView	
TrackId	**Nom**
1	Gens
2	Gens relacionats amb el càncer

Taula. 5.3: Relació de features amb el track amb el camp ParentId

També cal notar que en la versió optimitzada les diferents Features es relacionen amb un Track, i no amb un FeatureView com en la versió original. Així doncs el model resultant és el següent.

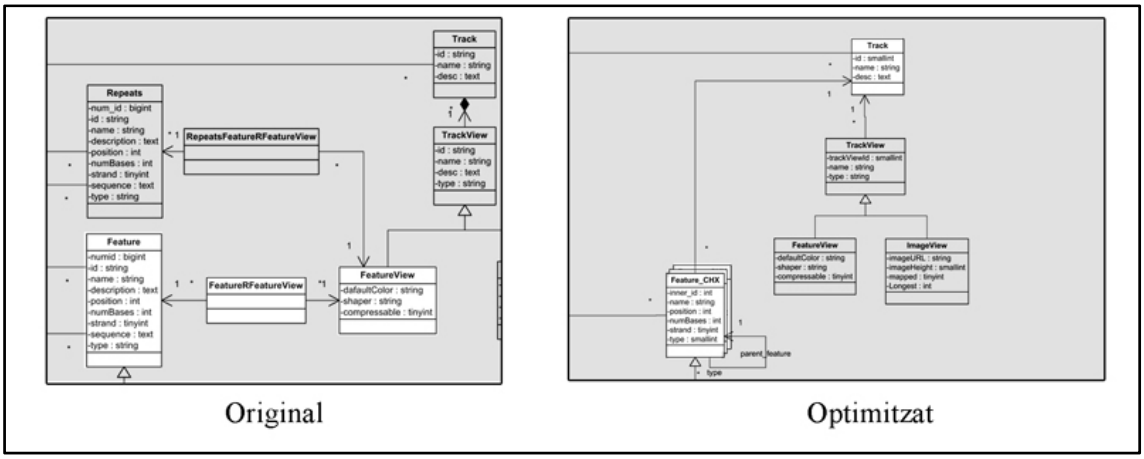


Fig 5.6: Eliminació de la classe FeatureRFeatureView

Taula TableDicc

Aquesta taula ens permet establir una relació entre el valor numèric que indica el tipus de la característica amb el nom de la taula que li correspon. D'aquesta manera les tuples que conté són de l'estil mostrat a continuació:

Num del tipus	Nom de la taula
1	Gene
2	Intron
3	Exon
4	Contig

Taula. 5.4: Relació de la taula TableDicc

5.1.2 Model lògic de la base de dades

A continuació es mostra el model lògic de la base de dades optimitzada fruit de la traducció del model conceptual que acabem de veure.

Taules d'organismes i cromosomes

Organism(id, name, desc, taxon_id)
Chromosome(id, orgId, name, desc, order), {on orgId referencia a Organism}

Taules de Tracks

Track(id, name, desc, orgId), {on orgId referencia a Organism}
TrackView(id, trackId, name, type), {on trackId referencia a Track}
FeatureView(trackViewId, defaultColor, shaper, compressable), {on trackViewId referencia a TrackView}
ImageView(trackViewId, imageURL, imageHeight, mapped, longest), {on trackViewId referencia a TrackView}

Taula de Features

Feature_CHX(inner_id, name, position, numBases, strand, type, trackId, parentFeature), {on trackId referencia a Track i parentFeature referencia a Feature_CHX}

Taules de les subclasses de Feature

Contig(chrId, featureId, desc, id_bd), { on featureId referencia a Feature_CH[chrId] }
Gene(chrId, featureId, geneId, symbol, pseudogene, HGNC, ensemblId, desc), {on featureId referencia a Feature_CH[chrId] }
Exon(chrId, featureId, pseudo, desc) {on featureId referencia a Feature_CH[chrId] }
Intron(chrId, featureId, desc), {on featureId referencia a Feature_CH[chrId] }
CodingExon(chrId, featureId, desc), {on featureId referencia a Feature_CH[chrId] }
miRNAHairpin(chrId, featureId, accession, HGNC, desc), {on featureId referencia a Feature_CH[chrId] }
SNP(chrId, featureId, rs, snpClass, snpType, seq5p, change, seq3p, avgHet, sdHet, validation, desc), {on featureId referencia a Feature_CH[chrId] }
Sts(chrId, featureId, uniSTS, desc, sequence, geneID), {on featureId referencia a Feature_CHX}
Repeat(chrId,featureId, desc, sequence),{ on featureId referencia a Feature_CH[chrId]}
MatureMicroRNA(chrId, featureId, accession, desc, sequence), {on featureId referencia a Feature_CH[chrId]}

Taules de configuració

ProfileParamaters(id, name, imageWidth)
ZoomLevels(id, basesPer1000px, order)
Densities(num_id, fearType, filePath, elemsPerFile, distance, maximumCount, chrId), {on chrId referencien a Chromosome}
TableDicc(tableNumber,tableName)

5.1.3 Tria del Sistema de Gestió de Bases de Dades

Com sabem el sistema original estava implementat amb MySQL. Aquest sistema gestor de base de dades es caracteritza per la seva velocitat però també ho fa per les seves limitacions, i és que a MySQL li manquen (o li mancaven) moltes de les característiques habituals d'un bon SGBD com el control de claus foranes, control de transaccions, disparadors... Una de les premisses a l'hora de plantejar-se utilitzar un nou SGBD és que aquest havia de ser de codi obert, aquest fet ens deixa principalment dos candidats: PostgreSQL i MySQL. A continuació analitzarem quins són els pros i els contres de cada un d'ells.

PostgreSQL

Pros

- Posseeix una gran escalabilitat, és capaç d'ajustar-se al número de CPUs i a la quantitat de memòria que posseeix el sistema de forma òptima, fent-lo capaç de suportar una major quantitat de peticions simultànies de manera correcta.
- Implementa l'ús de rollback's, subconsultes i transaccions, fent el seu funcionament molt més eficaç, i oferint solucions en camps en les que MySQL no podria. Té la capacitat de comprovar la integritat referencial, així com també d'emmagatzemar procediments a la pròpia base de dades, equiparant-lo als gestors de bases de dades d'alt nivell.

Contres

- Consumeix una gran quantitat de recursos.
- És de 2 a 3 vegades més lent que MySQL.

MySQL

Pros

- El millor de MySQL és la seva velocitat a l'hora de realitzar les operacions, el que el converteix un dels gestors que ofereixen major rendiment.
- Disposa d'eines d'administració molt bones, com phpMyAdmin que permeten tractar les dades de forma eficient.
- El conjunt d'aplicacions Apache-PHP-MySQL és un dels més utilitzats a Internet. Aquest fet fa que hi hagi molta documentació al respecte.

Contres

- No ofereix suport de transaccions, rollback's i subconsultes.
- No tracta la integritat referencial, és a dir, no permet implementar claus foranes, almenys en el seu motor més eficient (MyISAM).

Vistos els pros i els contres de cadascun podem concloure que PostgreSQL esdevé una bona solució per aquelles aplicacions que requereixen un SGBD capaç d'administrar restriccions i altres condicions de complexitat elevada però sacrificant-ne el rendiment. No obstant, si el que volem és un SGBD amb un alt rendiment del que no necessitem un control complex de les dades MySQL esdevé la millor opció.

Per a la nostra aplicació veiem que MySQL és el millor candidat a SGBD, ja que principalment volem rendiment en les lectures, ja que les escriptures són fetes off-line i mai pels usuaris del navegador, per tant no es fa necessari un control de les transaccions. Tot i que podria ser interessant poder implementar claus foranes s'ha decidit atorgar aquesta responsabilitat a la aplicació de control de dades. Un cop hem triat que utilitzarem MySQL com a SGBD hem de triar quin dels motors de base de dades que ofereix MySQL s'ajusta més als nostres requeriments. Amb l'ajut de la bibliografia hem determinat que el motor MyISAM és el que millor s'ajusta a les nostres necessitats ja que ofereix un alt rendiment per a aquelles taules en les que les escriptures són gairebé anecdòtiques en comparació amb les lectures. L'altre candidat, InnoDB, és menys eficient però ofereix noves funcionalitats com el control de claus foranes. Per tant, després del treball de comparació realitzat hem pogut determinar que el SGBD que utilitzarem per implementar la base de dades serà MySQL amb MyISAM com a motor de base de dades. Concretament s'utilitzarà la versió 5.0 d'aquest SGBD.

5.1.4 Quantificació del volum de dades

En aquest apartat determinarem de forma quantitativa quin és el volum de dades aproximat que es guardarà a la base de dades, això ens permetrà fer un disseny físic més acurat. Tot i que ja sabem per endavant que la quantitat de dades a emmagatzemar és elevada és interessant observar en quines classes s'hi guardarà més informació. Així doncs veiem, taula per taula, quin és el nombre de tuples esperat, aquest càlcul s'ha fet extraient dades del NCBI i amb l'ajuda del desenvolupador del navegador:

Nom de la taula	Nombre de tuples aproximat
Organisms	40
Chromosome	1600
Track	400
TrackView	800
FeatureView	400
ImageView	400
Densities	4800
ZoomLevels	10
ProfileParameters	10
TableDicc	20
Feature_CHX	3.000.000
Contig	15.000
Gene	2.000.000
Exon	10.000.000
Intron	10.000.000
CodingExon	100.000
MiRNAHairpin	50.000
SNP	60.000.000
STS	3.000.000
Repeat	10.000.000
MatureMicroRNA	50.000

Taula. 5.5: Nombre de tuples aproximat per taula

Així doncs, hem vist com les taules amb més tuples es corresponen a les taules Feature_CHX i algunes de les seves subclasses. En el disseny físic caldrà emfatitzar en

aquestes taules per tal de reduir al mínim la mida de les seves tuples així com de dissenyar unes estructures d'accés eficients.

5.1.5 Disseny físic de la base de dades

A partir del model lògic i amb el fet de que ja hem triat un SGBD ens podem plantejar quins tipus de dades utilitzarem per implementar els atributs de les diferents classes. Tal i com hem dit l'objectiu és minimitzar al màxim la mida de les tuples, tot i així, els tipus de dades han de ser els adequats per poder treballar amb el rang de valors requerit per l'atribut. MySQL ofereix un bon nombre d'opcions a l'hora de triar un tipus de dades. A continuació descrivim les característiques dels més rellevants i que són requerits pel nostre model de dades.

Dades numèriques

En funció del rang de valors desitjat podem utilitzar els següents tipus de dades que són implementats amb un nombre variable de bits:

- TINYINT (8 bits)
- SMALLINT (16 bits)
- MEDIUMINT (24 bits)
- INT (32 bits)
- BIGINT (64 bits)

Per tant, el rang de valors enters que ofereix cada un d'ells va des de $-2^{(N-1)}$ fins a $2^{(N-1)}-1$. Tot i així, ens pot interessar optar per valors naturals, ja que podem no requerir valors negatius, d'aquesta manera dupliquem el rang de valor positius disponibles.

Dades alfanumèriques

MySQL ofereix els tipus alfanumèrics CHAR i VARCHAR, en funció de si volem longitud fixa o longitud variable. En ambdós casos hem d'especificar una longitud màxima per a la cadena, però mentre que CHAR sempre utilitza tots l'espai especificat, VARCHAR utilitza només la quantitat de caràcters que es necessitin per a aquell valor. A més, MySQL ofereix el tipus TEXT per a guardar cadenes molt llargues.

A l'annex 3 hi podem veure la relació de quins han estat els tipus de dades usats en les classes més rellevants de la base de dades.

També forma part del disseny físic el treball d'elegir les estructures d'accés que utilitzarem en cada una de les classes. El motor MyISAM ofereix l'índex B-Tree, la idea general d'un arbre B-Tree és que tots els valors estan guardats en ordre, i cada pàgina fulla es troba a la mateixa distància de l'arrel. Un arbre B-Tree accelera l'accés a les dades perquè el motor d'emmagatzemament no ha de escanejar la taula sencera per trobar les dades desitjades. En comptes d'això es comença pel node arrel, aquest conté punters cap als nodes fills, i el motor d'emmagatzemament segueix aquests punters. Agafa el punter adequat mirant els valors emmagatzemats als nodes, que marquen els límits superiors i inferiors dels valors dels nodes fills. Les pàgines filles són especials perquè contenen punters al espai lògic on es troben les dades emmagatzemades. Degut a

que les B-Trees guarden les columnes indexades en ordre, són útils per buscar per rangs de valors. És per això que és interessant utilitzar B-Trees per a la nostra aplicació, ja que com hem vist, ens pot interessar buscar les característiques que es troben entre dues posicions determinades.

Veiem que per totes les classes, excepte la classe Feature_CHX, en tenim prou amb crear l'índex per clau primària, que de fet MySQL crea automàticament, però amb la classe Feature_CHX ens pot interessar, a més, indexar per 'Position' i per 'trackId', ja que són paràmetres usats en la cerca. Així doncs de moment creem aquests índexs a la taula Feature_CHX.

5.1.6 Definició final del model de la base de dades

Amb la definició del model lògic i físic hem pogut definir un model final de la base de dades. Aquest model consta ja de les sentències SQL per a la creació de les taules, les quals incorporen la creació dels índexs. Podem trobar aquestes sentències de creació de les taules a l'annex 4. Aquestes sentències són les que carregarem al servidor MySQL per tal de crear-hi la base de dades.

5.1.7 Disseny d'una arquitectura de servidors

En aquest apartat es vol donar resposta a un dels objectius del projecte que consisteix en dissenyar un esquema de servidors de bases de dades que sigui capaç d'administrar un alt nombre de peticions de dades. Tot i que ara per ara no es fa necessari implementar un sistema d'aquestes característiques per a Genexp és adequat planificar-lo per quan sigui necessari.

Hi ha vàries solucions arquitectòniques que ofereixen escalabilitat i alta disponibilitat, una d'elles és el clustering, que consisteix en dividir la informació en diferents màquines i dirigir la petició a la màquina que li correspon. En el nostre cas es podria arribar a assignar una màquina per cada cromosoma o subtipus d'informació genètica. Això implicaria tenir un excessiu nombre de màquines al servei de l'aplicació, per tant no es considera una solució adequada.

En comptes del clustering sembla molt més eficient aplicar l'estratègia del balanceig de càrrega. Aquesta arquitectura consisteix en la utilització d'uns sistemes específics anomenats balencejadors de càrrega que distribueixen les peticions entre els servidors en funció de la seva càrrega. D'aquesta manera s'obté una major eficiència del sistema ja que l'objectiu és intentar que tots els servidors s'utilitzin en la mateixa proporció. Aquesta arquitectura implica que tots els servidors de bases de dades han de tenir la mateixa informació replicada. En el cas de Genexp això no resulta especialment un problema ja que es tracta d'una aplicació de lectura massiva i les escriptures es fan de forma centralitzada per un sistema de gestió de les dades. A la figura 5.7 podem veure aquesta proposta arquitectònica.

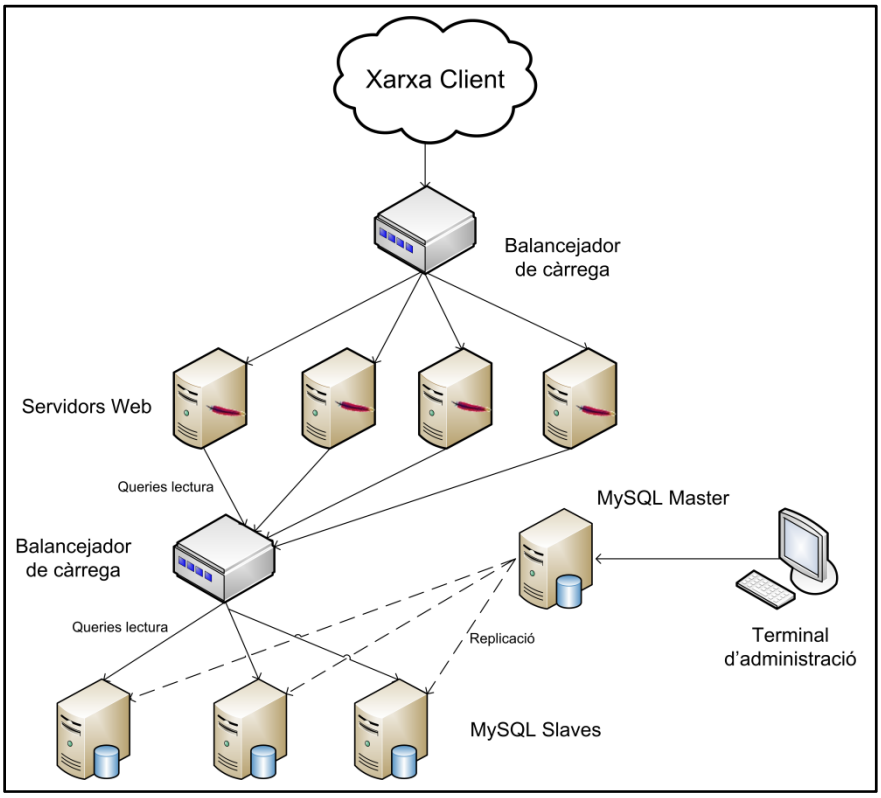


Fig 5.7: Disseny de l'arquitectura de servidors

Per donar suport a la replicació de dades cal que existeixi una màquina que actua de Master on s'hi realitzen les escriptures i aquesta les propaga als diferents Slaves que són els que donen resposta a les lectures. En aquest disseny, a més de distribuir l'accés a la base de dades també s'ha balancejat l'accés a l'aplicació web. Amb aquesta arquitectura es pot obtenir un alt rendiment i una alta disponibilitat.

5.2 Resultats obtinguts

Per mitjà de tests hem avaluat quin és el rendiment de la base de dades. L'estructura dels tests és la mateixa que en l'apartat 4.4 de la memòria ja que així podem fer una comparació adequada entre el sistema original i l'optimitzat. Com que ha canviat el model de dades també ho ha fet la crida SQL per obtenir les dades, aquesta és la que mostrem a continuació:

```
SELECT innerId, name, position, numBases, strand, type
FROM Feature_chr18
WHERE trackId=2 AND position BETWEEN 15451 AND 15465211
ORDER BY position;
```

Test 1: Obtenció de característiques de tipus Feature View

En aquest test veiem el comportament del sistema a l'hora de retornar tracks del tipus FeatureView que són tracks amb un nombre relativament baix de característiques.

En aquest test farem varies proves en funció de diferents FeatureViewID; realitzarem crides en els tracks de contigs, gens i STS mentre que deixarem fix el cromosoma.

Track dels contigs:

S'han fet 100 repeticions de la crida que ens retorna els contigs d'un cromosoma determinat, en aquest cas del 18. Els resultats obtinguts són els següents:

Variable estadística	Valor BD optimitzada	Valor BD original
Mitjana del temps de resposta	0,18	0,14 s
Desviació estàndard del temps de resposta	0,07	0,02 s

Taula 6.1: Resultats del test 1, track dels contigs

Track dels gens:

En aquest cas s'han fet 100 crides per obtenir els gens del cromosoma 18. Els resultats són els següents.

Variable estadística	Valor BD optimitzada	Valor BD original
Mitjana del temps de resposta	0,18	0,72 s
Desviació estàndard del temps de resposta	0,05	0,1 s

Taula 6.2: Resultats del test 1, track dels gens

Track dels STS:

Finalment hem fet 100 repeticions de la crida que retorna els STS del cromosoma 18. A continuació veiem els resultats.

Variable estadística	Valor BD optimitzada	Valor BD original
Mitjana del temps de resposta	0,31	7,28 s
Desviació estàndard del temps de resposta	0,05	0,14 s

Taula 6.3: Resultats del test 1, track dels sts

Podem veure com hem aconseguit un guany significatiu tant pel que fa als gens però sobretot pel que fa als STS. No obstant pel que fa als contigs podem afirmar que els resultats són similars als originals. En les gràfiques següents podem veure aquest guany de forma visual:

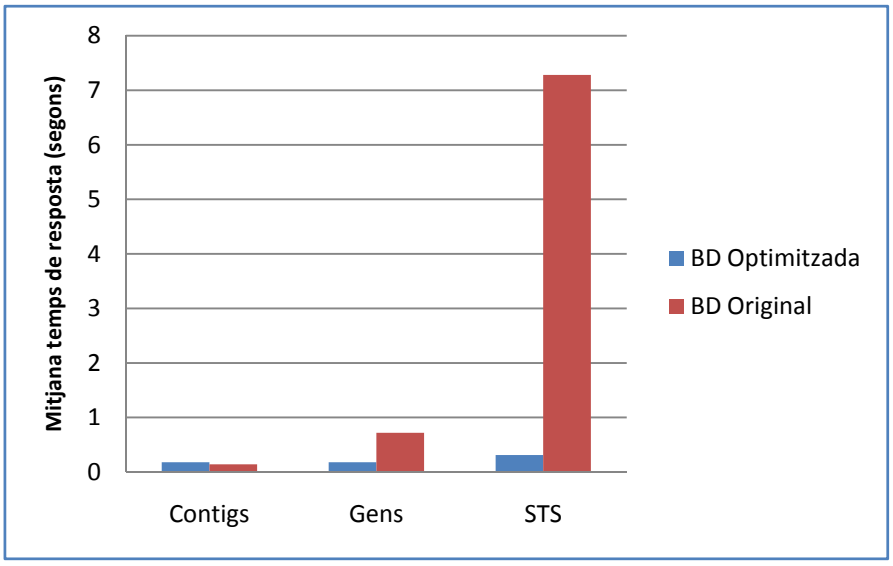


Fig 6.1: Guany en la recuperació de les característiques de tipus feature View

Test 2: Obtenció de característiques de tipus Image View

L'objectiu d'aquest test és veure com es comporta el sistema a l'hora de retornar característiques d'un track de tipus Image View. S'ha deixat fix el track (en aquest cas s'ha elegit el track dels SNPs) i el cromosoma (s'ha elegit el cromosoma 18 del Homo Sapiens) i s'ha anat variant aleatòriament el rang de posicions.

Després de fer 1000 repeticions d'aquesta prova hem obtingut els següents resultats:

Variable estadística	Valor BD optimitzada	Valor BD original
Mitjana del temps de resposta	2,81 s	23,38 s
Desviació estàndard del temps de resposta	1,62 s	11,92 s

Taula 6.4: Resultats del test 2

En el gràfic següent podem veure els resultats d'aquest test, també es mostren els resultats de la base de dades original per poder observar el guany aconseguit.

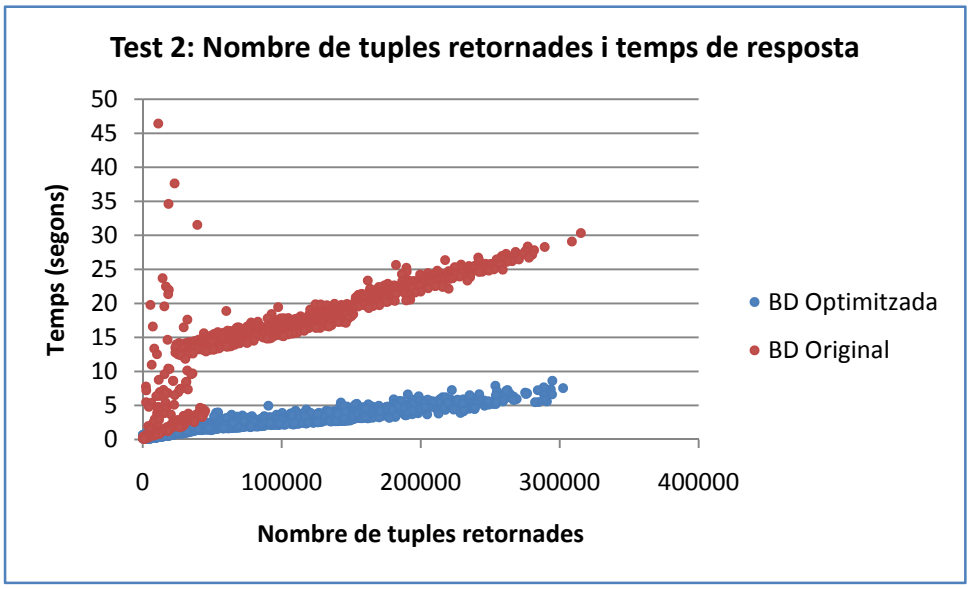


Fig 6.2: Resultat del test 2 comparant el nombre de tuples retornades amb els temps de resposta obtinguts en les dues bases de dades

En aquest gràfic podem veure com els temps de la base de dades optimitzada, tot i que presenten una tendència alcista, són més estables que la base de dades original.

En el següent gràfic veiem de forma visual el guany en la mitjana del temps de resposta.

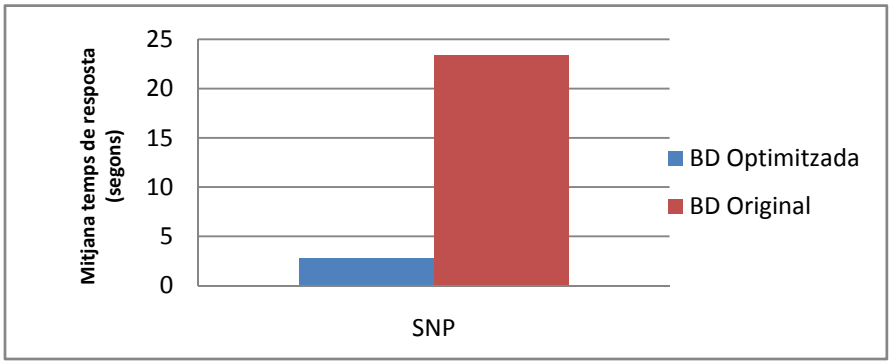


Fig 6.3: Mitjanes dels temps de resposta a l'hora de retornar els SNPs

6 Desenvolupament de l'aplicació de control

Tal i com hem explicat, un dels objectius del projecte és implementar una aplicació que ens permeti fer un control de les dades que hi ha a la base de dades. Aquesta aplicació ha de ser capaç de descarregar els fitxers d'informació genòmica d'Internet, parsejar-los i carregar les dades a la base de dades. En aquest apartat veurem quines han estat les decisions preses en el disseny d'aquesta aplicació així com els resultats del testejat de l'aplicació.

6.1 Disseny de l'aplicació de control

Veiem quines han estat les decisions preses durant el procés d'elaboració de l'aplicació de control de les dades.

6.1.1 Definició dels casos d'ús

En aquest apartat descriurem els diferents casos d'ús de l'aplicació de control de dades, aquests casos d'ús han de satisfer els requeriments descrits en el punt 3.3 de la memòria. En tots els casos d'ús d'aquesta aplicació només hi ha un actor implicat que esdevé l'administrador de les dades genòmiques de Genexp.

Casos d'ús de l'aplicació general

L'aplicació general ha de permetrer realitzar totes les accions que no siguin específiques d'una font genòmica determinada, és a dir, ha de poder gestionar els organismes, així com els seus cromosomes o els tracks. A més a més ha de poder gestionar les diferents bases de dades existents en el sistema.

Gestió d'organismes:

Aquest cas d'ús consta de les funcionalitats que permeten gestionar els diferents organismes existents al sistema.

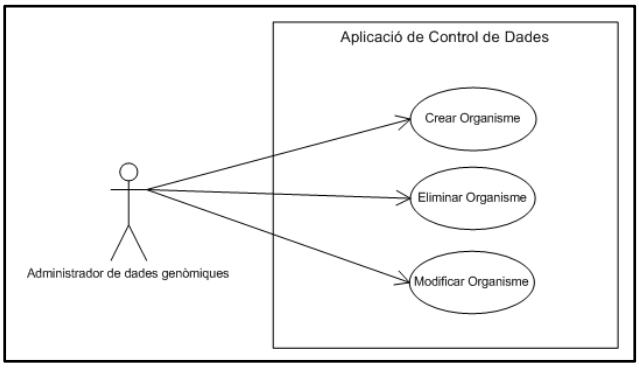


Fig 6.1: Casos d'ús de la gestió d'organismes

Gestió de cromosomes:

Amb aquest cas d'ús podem gestionar els cromosomes dels organismes.

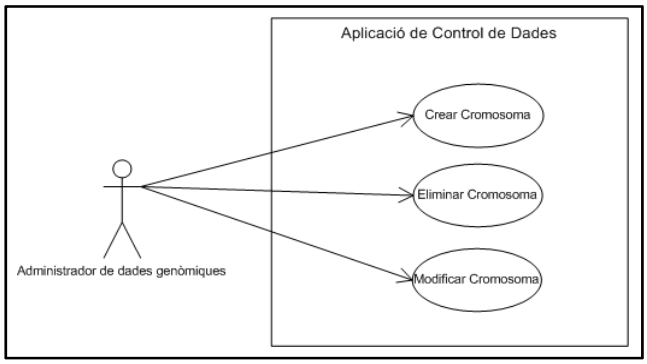


Fig 6.2: Casos d'ús de la gestió de cromosomes

Gestió de Tracks:

Gràcies a aquest cas d'ús podem gestionar els diferents tracks, així com especificar-ne les seves característiques en funció de si es tracten de FeatureViews o TrackViews.

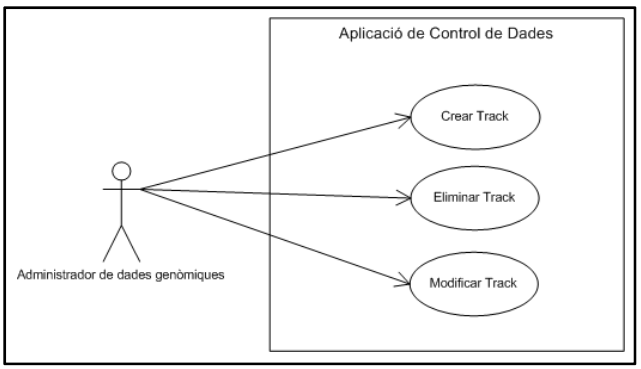


Fig 6.3: Casos d'ús de la gestió de cromosomes

Gestió de Bases de Dades:

L'objectiu d'aquest cas d'ús és poder gestionar les diferents versions de bases de dades existents en el sistema. Recordem que és necessari que hi hagi 3 versions diferents de la base de dades: una que és la que està en producció, una que és una còpia d'aquesta i una altra que és una versió antiga de la base de dades. L'aplicació ens ha de permetre canviar el rol jugat per aquestes bases de dades.

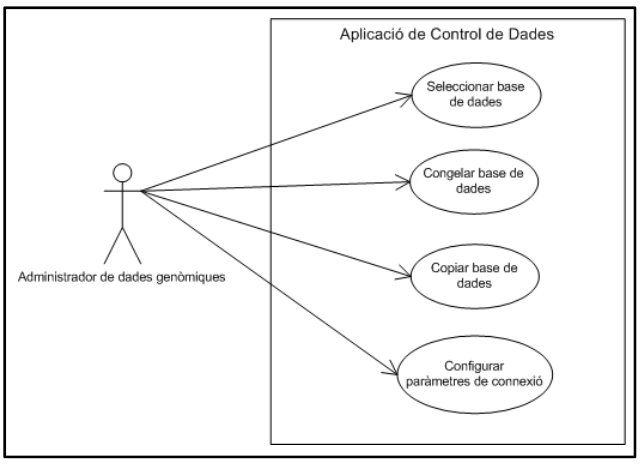


Fig 6.4: Casos d'ús de la gestió de bases de dades

Casos d'ús de la sub-aplicació del parser de Genbank

Recordem que per cada font d'informació genòmica hi ha d'haver una sub-aplicació específica que sigui capaç de tractar-ne les dades. Aquestes sub-aplicacions s'han d'integrar amb l'aplicació general. Tal i com hem vist en la definició de l'abast del projecte, només s'ha implementat la sub-aplicació que tracta les dades de la font Genbank. A continuació veiem quins són els seus casos d'ús.

Gestió de les dades existents:

Amb aquest conjunt de casos d'ús es pretén que l'usuari pugui obtenir informació sobre l'estat de les dades existents, així com actualitzar-les mitjançant la descàrrega de nova informació genòmica.

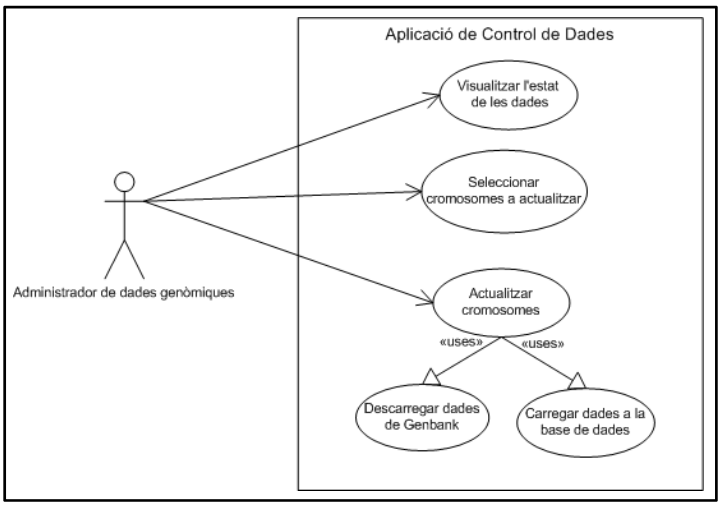


Fig 6.5: Casos d'ús de la gestió de dades existents

Gestió de les configuracions dels organismes, els cromosomes i els tracks:

L'aplicació ha de permetre configurar alguns paràmetres dels organismes, com per exemple l'adreça on es troba la seva informació genòmica, també ha de permetre associar els tipus de dades de Genbank als tracks existents.

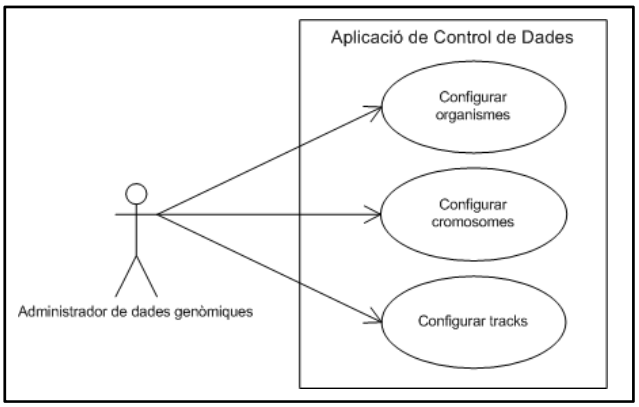


Fig 5.10: Casos d'ús de la configuracions d'organismes, cromosomes, i tracks

6.1.2 Disseny de l'arquitectura de l'aplicació

Tal i com hem vist és interessant que l'aplicació consti d'una aplicació general que ens permeti gestionar informació independent de les fonts de dades genòmiques i de vàries aplicacions independents que gestionin la informació específica de les fonts de dades genòmiques. El disseny escollit per donar resposta a aquesta necessitat consisteix en aplicar un sistema de plug-ins on es poden afegir noves sub-aplicacions de forma dinàmica sense haver de re-compilar l'aplicació, d'aquesta manera cada una de les sub-aplicacions específiques esdevindrà un plug-in de l'aplicació general.

S'ha optat per fer servir un model de tres capes com a arquitectura de l'aplicació, en aquest model hi ha una capa de presentació que manté la comunicació amb l'usuari, una capa de domini que gestiona tota la lògica de l'aplicació i una capa de dades que s'encarrega de gestionar les dades de l'aplicació. Amb aquest model s'aconsegueix:

- Que un canvi en la representació persistent de les dades (per exemple, un canvi en el sistema gestor de bases de dades o fitxers), normalment, només afecti a la capa de gestió de dades.
- Que un canvi en la interfície del programa afecti només la capa de presentació.
- Que la capa de domini, que encapsula la majoria de la lògica del programa, sigui força independent dels canvis de plataforma, sistema operatiu, etc.

Veiem ara algunes decisions de disseny preses en cadascuna d'aquestes tres capes.

Capa de presentació

S'ha optat per mostrar un diàleg específic per a cada cas d'ús, d'aquesta manera obtenim una aplicació més intuïtiva i fàcil de fer servir.

Quan iniciem l'aplicació ens apareixerà el diàleg que ens ha de permetre gestionar les diferents bases de dades. En aquest diàleg hi apareixerà una il·lustració que ens mostri quin rol està jugant cada base de dades, a més ens ha de permetre accedir a la configuració dels paràmetres de configuració de la base de dades com el username o el password. A continuació veiem un esbós de com hauria de ser aquest diàleg:

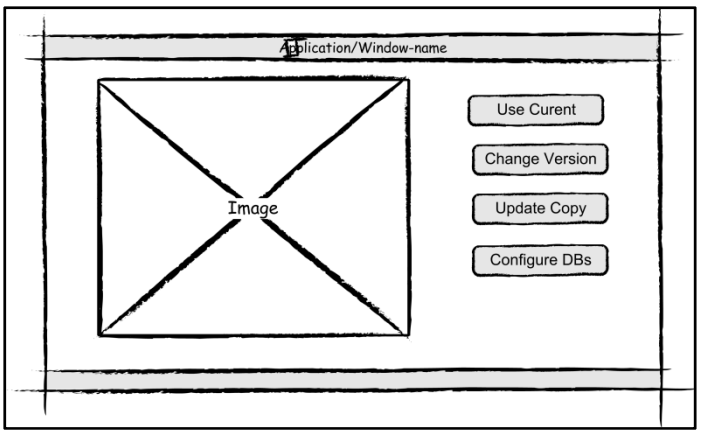


Fig 5.11: Esbós del diàleg de configuració de la base de dades

Un cop hem seleccionat la base de dades accedim a l'aplicació principal de gestió de dades. En aquest diàleg hem de poder accedir a les funcionalitats de l'aplicació de dades general. També hem de visualitzar els marcs dels diferents plug-ins. S'ha optat per a que aquests apareguin d'uns d'un marc amb pestanyes mentre que les característiques de l'aplicació general apareguin en una barra d'icones a la part superior.

Ho veiem a la següent figura:

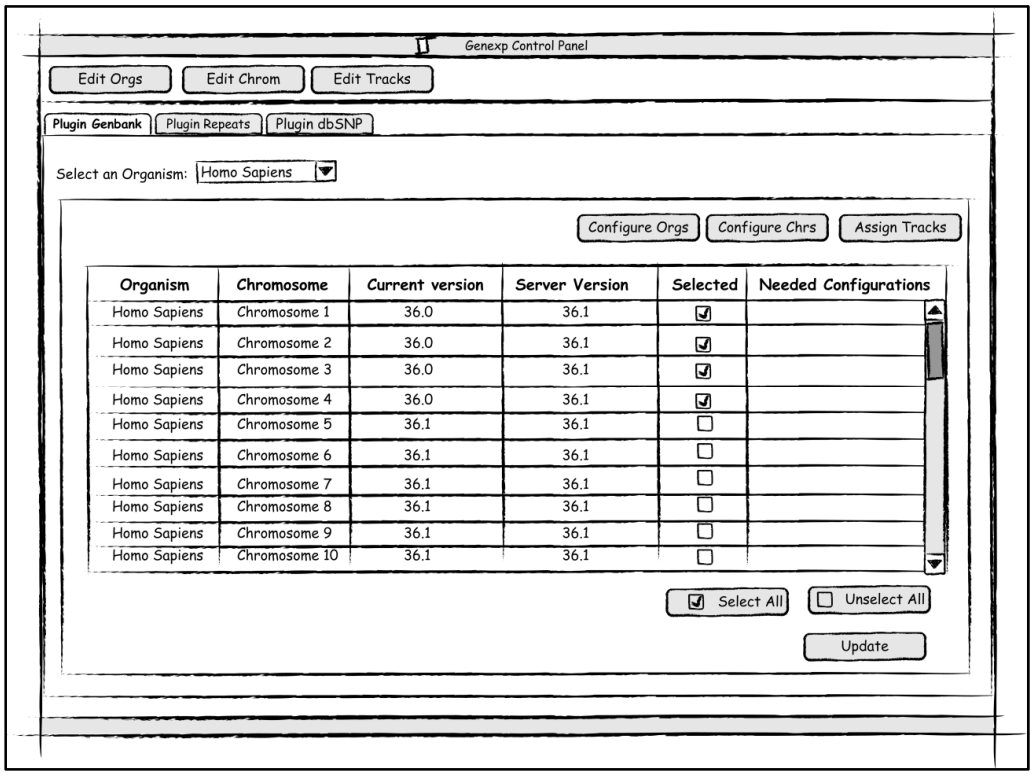


Fig 5.12: Esbós del diàleg de l'aplicació general amb el plug-in de Genexp

A través dels botons "Edit Orgs", "Edit Chrom" i "Edit Tracks" hem de poder accedir als formularis que ens permeten crear i visualitzar nous organismes, cromosomes i tracks.

Pel que fa al marc específic del plug-in de Genbank hi ha d'aparèixer una taula amb una fila per cada cromosoma del organisme que s'ha seleccionat en una llista. En cada una d'aquestes files hi ha d'aparèixer el nom del cromosoma, la versió que hi ha guardada a la base de dades, la versió que hi ha al servidor de Genbank i un indicador que ens avisi de si falta configurar algun paràmetre (com per exemple que no hi hagi especificada l'adreça del fitxer de dades). Hem de poder seleccionar els cromosomes que volem actualitzar i a continuació demanar a l'aplicació que els actualitzi. En aquest moment s'ha de mostrar un diàleg com el que es mostra a la figura 5.12. En aquest diàleg hi han d'aparèixer els cromosomes que es volen actualitzar i s'ha de mostrar l'estat del procés d'actualització en què es troba cada un d'ells.

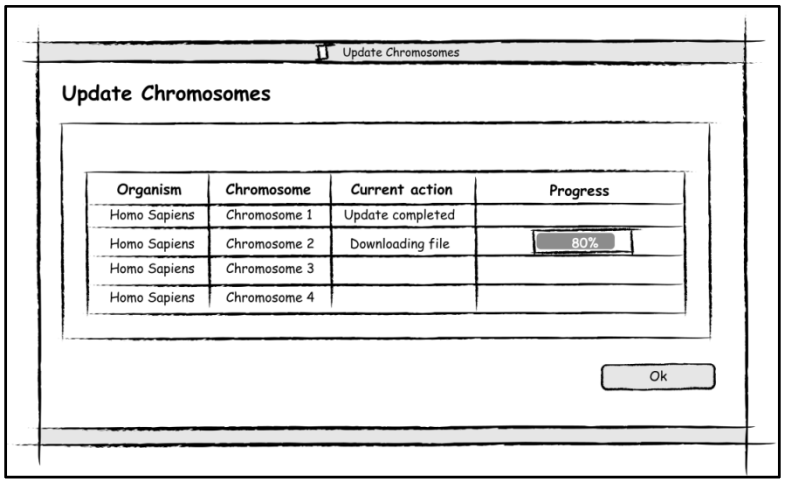


Fig 5.12: Esbós del diàleg d'actualització de cromosomes

Capa de domini

S'ha decidit emprar el model controlador façana com a patró de disseny per gestionar el domini de l'aplicació. En aquest model es centralitzen totes les peticions a una mateixa classe. S'ha optat per aquest patró perquè el nombre de possibles accions efectuades per l'usuari no és massa elevat.

Així doncs hi ha una classe Controlador que gestiona tota la lògica de l'aplicació, val a dir que aquesta capa té poques responsabilitats ja que la major part de la feina és efectuada per la capa de gestió de dades. Per tant aquesta capa es limita bàsicament a comunicar la capa de presentació amb la capa de gestió de dades.

Capa de gestió de dades

És important remarcar que en aquesta capa s'han pres algunes decisions de disseny importants:

L'aplicació principal disposa del seu propi controlador de dades, aquest s'encarrega de fer la connexió amb la base de dades i de fer-hi les crides pertinents. Aquest controlador també actua de passarel·la de cara a la les diferents sub-aplicacions, ja que incorpora funcions que permeten efectuar operacions comunes a totes elles com pot ser la de "Inserir Característica", la qual insereix una tupla a la superclasse Feature_CHX, mentre que inserir la tupla a la subclasse és responsabilitat del controlador extès.

Les sub-aplicacions interaccionen amb la base de dades a través d'un controlador que és una extensió del de l'aplicació principal i que utilitza la seva connexió amb la base de dades. D'aquesta manera s'aconsegueix independència entre les diferents sub-aplicacions i amb l'aplicació general, ja que els requeriments de dades específics (com les dades de les subclasses de Feature) els efectua el controlador del plug-in.

La informació de configuració requerida pels diferents plug-ins (com per exemple, la URL on es troben els fitxers de dades genòmiques de Genbank) es guarda en fitxers XML i no a la base de dades, així no es sobrecarrega aquesta i s'obté més

independència entre sub-aplicacions. Per llegir i escriure aquests XML les sub-aplicacions consten de petits controladors que gestionen aquestes dades de configuració. També cal recordar que, com hem vist en el disseny de la base de dades, aquesta no pot gestionar la integritat referencial, i que era responsabilitat de l'aplicació de control assegurar-la. És per això que el controlador implementa esborrats en cascada. Per exemple, si s'esborra un organisme de la base de dades caldrà esborrar-ne els cromosomes i tracks associats i , finalment, haurem d'esborrar les seves features conjuntament amb la informació guardada a les subclasses. Cal destacar que la comunicació entre les capes de domini i de dades es fa mitjançant classes que actuen d'embolcall de les taules existents a la base de dades, d'aquesta manera, la funció *retornaOrganisme(idOrganisme)* del controlador de dades retorna un objecte *Organisme* que té per atributs camps existents a la taula *Organism* de la base de dades.

6.1.3 Entorn i tecnologies usades

L'aplicació ha estat desenvolupada íntegrament en Java 1.6 i s'ha usat el programari Netbeans 6.5 com a entorn de desenvolupament integrat. Les interfícies gràfiques han estat implementades en Swing usant les eines que ofereix Netbeans. S'ha elegit aquesta tecnologia perquè ofereix una alta portabilitat entre sistemes operatius que és un dels requeriments no funcionals especificats, a més a més, durant tota la carrera hi he adquirit més experiència que amb d'altres llenguatges de programació.

D'altres tecnologies utilitzades són les que es descriuen a continuació:

- Per tractar els XML que guarden els paràmetres de configuració dels plug-ins s'ha usat l'API SAX (Simple API for XML) ja que és bastant fàcil d'utilitzar i no es fa necessari utilitzar cap llibreria addicional ja que està integrada en el JDK estàndard.
- Per a poder parsejar els arxius de Genbank s'ha usat la llibreria Biojava 1.6, aquesta llibreria ens permet obtenir les característiques incloses en aquests fitxers de forma bastant simple usant iteradors i una API específica.
- Per a descarregar els arxius de Genbank s'utilitza una llibreria FTP que ens permet connectar-nos al servidor i descarregar els arxius monitoritzant-ne el procés.
- Per a connectar-se a la base de dades MySQL l'aplicació usa el driver estàndard de MySQL, concretament, en la seva versió 5.1.

6.1.4 Sistema de plug-ins

Per gestionar les diferents sub-aplicacions, o plug-ins, que han de tractar les diferents fonts d'informació genòmica s'ha usat la llibreria Java Plug-in Framework (JPF). Aquesta llibreria ens permet programar plug-ins independentment i incorporar-los a l'aplicació de forma dinàmica ja que la detecció d'aquests es fa en temps d'execució i no en temps de compilació. A continuació veurem els aspectes més rellevants de l'aplicació d'aquesta tecnologia.

JPF utilitza un model en el que totes les parts del programa, inclosa l'aplicació principal, són plug-ins. A més d'aquests plug-ins hi ha un fitxer de configuració i una classe d'arrencada, en aquest fitxer s'hi indica quin és el plug-in que la classe d'arrencada ha d'iniciar en primera instància. Un cop hem iniciat aquest plug-in, que en el nostre cas és l'aplicació general, podem iniciar altres plug-ins, que en el nostre cas esdevenen les diferents fonts d'informació. Per a poder iniciar aquests plug-ins hem de definir el que "anomenem punt d'extensió". Aquest punt d'extensió consisteix en la definició d'una interfície *"Source"* que els diferents plug-ins implementen. Podem veure el funcionament del sistema de plug-ins a la figura 5.13.

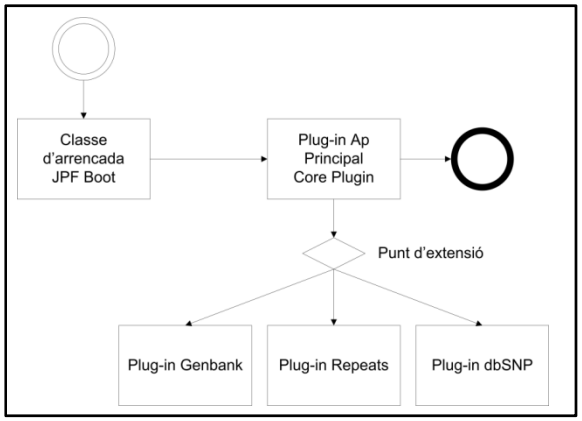


Fig 5.13: Flux del sistema de plug-ins

Cada un dels plug-ins pot incorporar les seves pròpies llibreries o utilitzar les de l'aplicació general. D'aquesta manera obtenim independència però també una alta integració dels components de l'aplicació. Aquests paràmetres de configuració s'especifiquen en el fitxer *Plugin.xml* que trobem en el directori de cada un dels plug-ins.

6.1.5 Descripció dels casos d'ús més representatius

En aquest punt descriurem mitjançant diagrames de seqüència UML el comportament de l'aplicació en alguns dels casos d'ús més rellevants.

Afegir Organisme

Aquest cas d'ús consisteix en afegir un organisme a la base de dades, veiem el diagrama de seqüència corresponent.

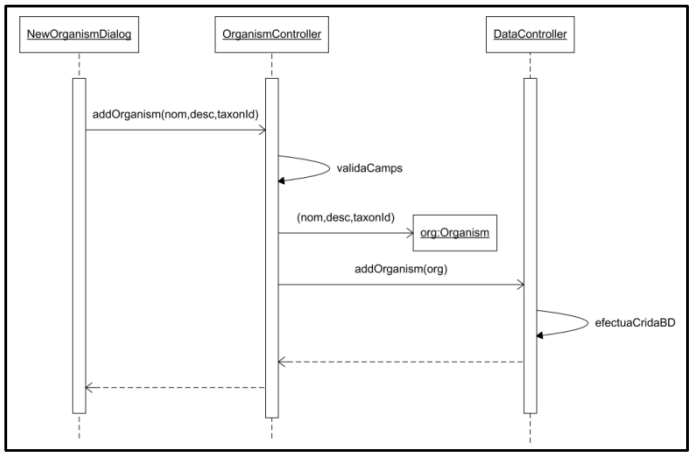


Fig 5.14: Diagrama de seqüència de afegirOrganisme

Veiem com el diàleg NewOrganism crida el controlador del domini corresponent que valida els camps, crea l'embolcall i crida al controlador de dades per a que afegeixi l'organisme a la base de dades.

Afegir Cromosoma

Aquest cas d'ús consisteix en afegir un cromosoma a la base de dades, a continuació podem veure el seu diagrama de seqüència associat:

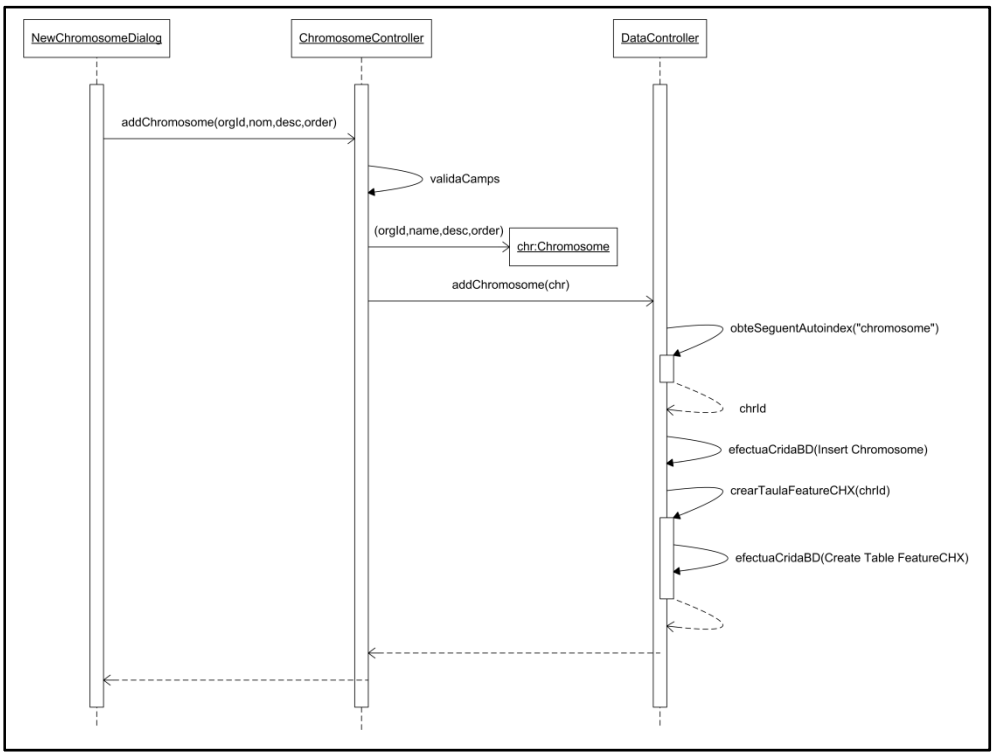


Fig 5.15: Diagrama de seqüència de afegirCromosoma

En aquest cas és necessari que el controlador de dades, a més d'afegir la tupla a la taula *Chromosome*, creï la taula *FeatureCHX* corresponent. Com que el nom de la taula ha de contenir l'identificador del cromosoma, cal que l'obtinguem mitjançant la crida *obteSeguentAutoindex* que consulta l'esquema de la base de dades per saber quin és el següent autoíndex de la taula *Chromosome.*

Eliminar organisme

A la figura 5.16 podem veure el diagrama de seqüència del cas d'ús eliminar organisme. La complexitat d'aquest diagrama és deguda al fet de que cal fer l'esborrat en cascada de totes les tuples associades a l'organisme que volem eliminar. Això implica, per una banda, esborrar els cromosomes associats a l'organisme, fet que, al seu temps, suposa eliminar les característiques que contenen, per tant caldrà esborrar la taula Feature_CHX del cromosoma i totes les tuples de les subclasses que corresponguin. Per una altra banda, cal esborrar també els tracks associats a l'organisme, això suposa haver d'esborrar les tuples de la taula Track, les tuples de la taula TrackView que hi estiguin relacionades i, finalment les tuples de les seves subclasses FeatureView i ImageView.

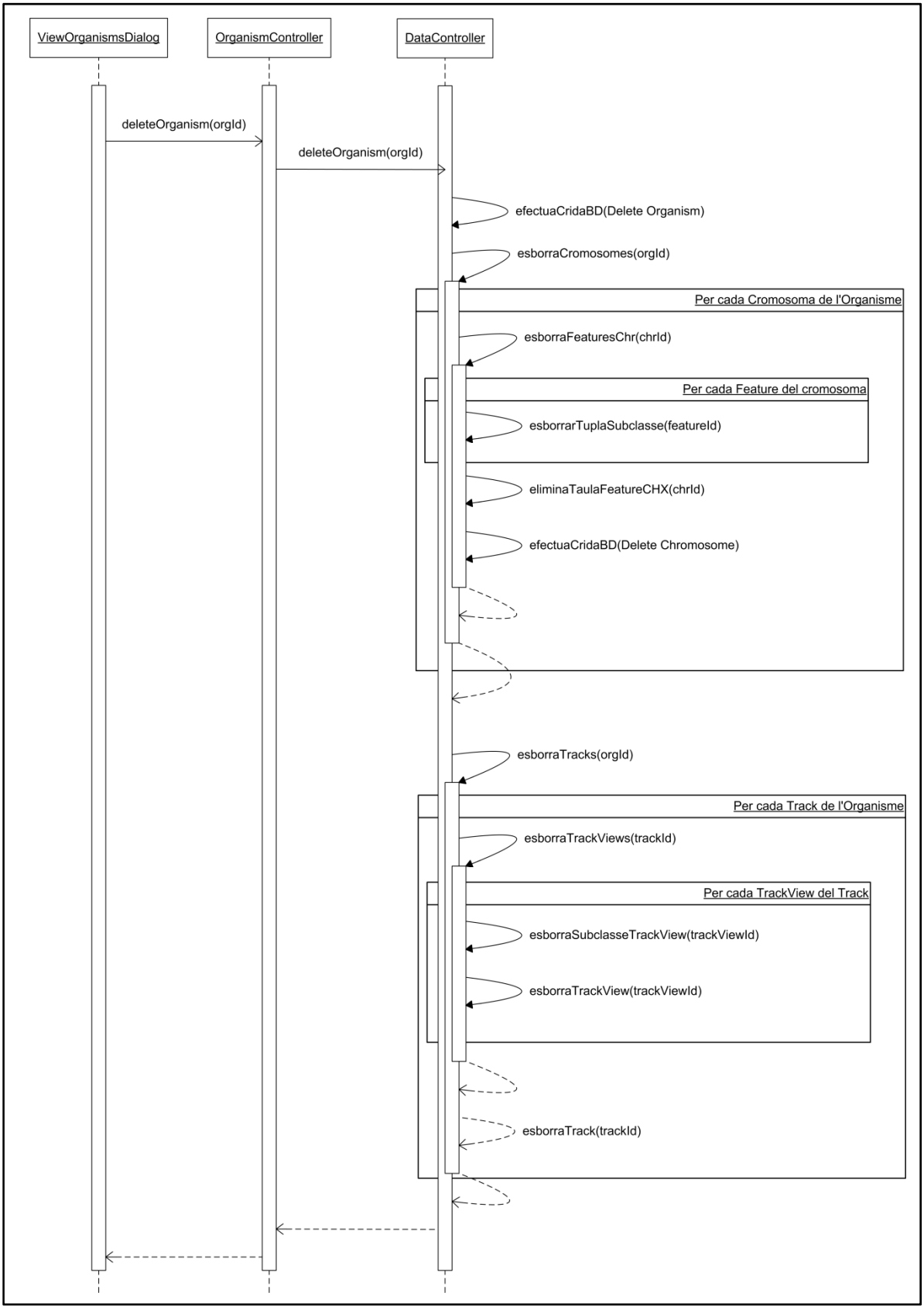


Fig 5.16: Diagrama de seqüència d'eliminar organisme

Actualitzar cromosoma

Aquest cas d'ús descriu el cas en el que volem actualitzar un cromosoma en el plug-in de Genbank, per tant l'haurem de descarregar, descomprimir-lo, parsejar-lo i incorporar-lo a la base de dades. També caldrà, abans de realitzar aquestes accions, esborrar les dades antigues.

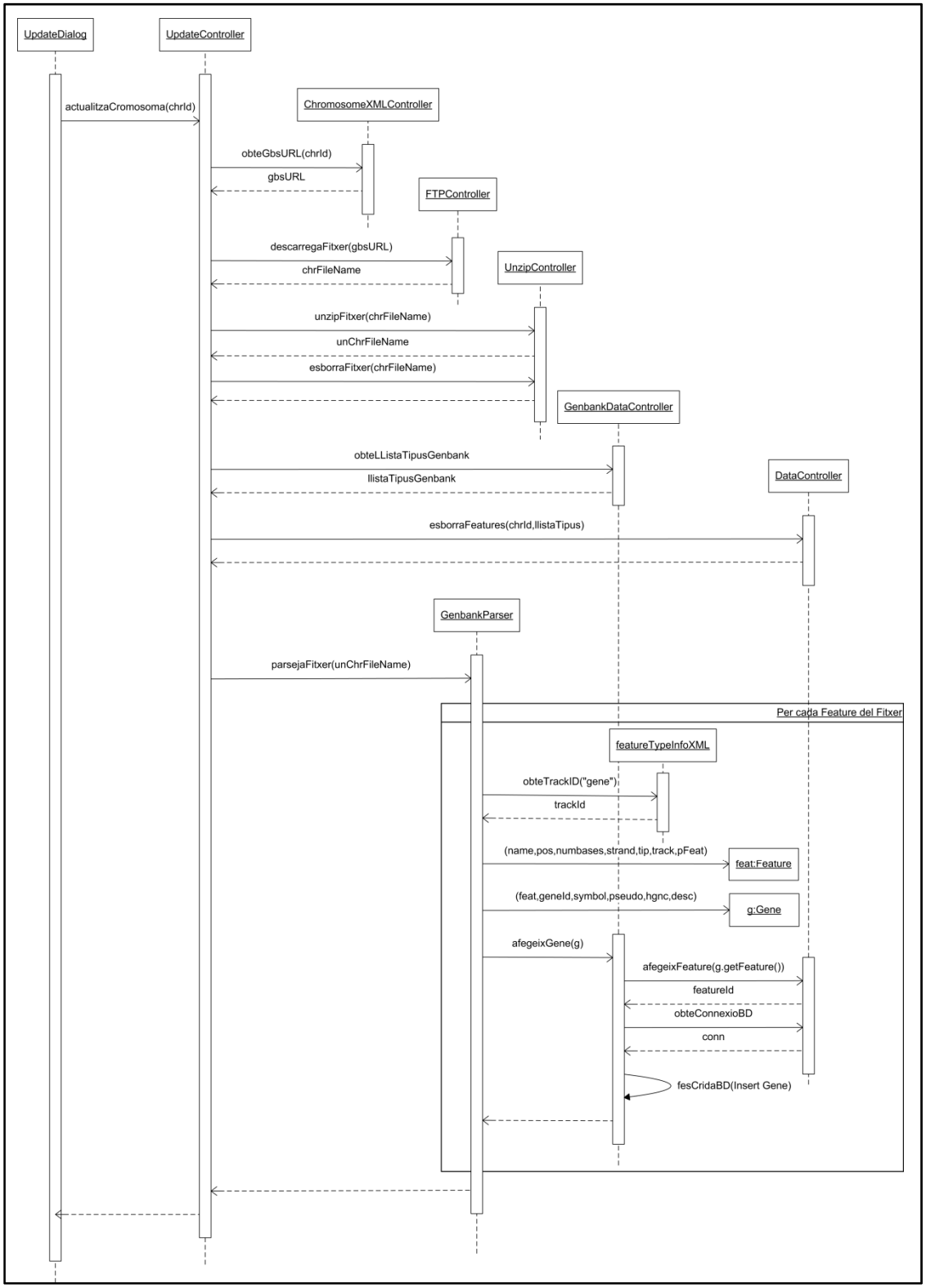


Fig 5.17: Diagrama de seqüència de actualitzar cromosoma

En el diagrama podem veure les accions que es realitzen per actualitzar cada un dels cromosomes que es volen actualitzar. Primer de tot s'ha de llegir el XML de configuració per obtenir la URL on es troba el fitxer de dades genòmiques de Genbank que correspon al cromosoma. A continuació cridem al controlador de FTP per descarregar el fitxer i guardar-lo en una directori de dades temporals. Després cal descomprimir-lo ja que originalment es troba comprimit. Seguidament cal esborrar les dades antigues, per tant cal esborrar les Features dels cromosoma que es corresponguin amb els tipus de característiques que ens disposem a actualitzar. Finalment passem el fitxer de dades al parsejador i per cada Feature l'afegim a la base de dades. En el diagrama de seqüència només es mostren la inserció de gens, caldria fer aquesta operació per cada un dels tipus que es troben en el fitxer de Genbank.

Un fet que cal remarcar és que tot i que les característiques es troben ordenades per posició en el fitxer de Genbank, el parsejador les insereix ordenades per tipus, d'aquesta manera les característiques dels diferents tipus de dades genòmiques es guarden de forma consecutiva. Així aconseguim que les dades d'un mateix tipus estiguin juntes a disc i minimitzem el temps d'accés quan fem una cerca per visualitzar un tipus de dades genòmiques determinat.

6.2 Resultats obtinguts

A continuació mostrarem els resultats dels tests de funcionalitat que s'han fet sobre l'aplicació de control de dades implementada. Analitzarem el seu funcionament i veurem si satisfà els diferents requeriments. A l'annex 5 hi trobem el manual d'usuari de l'aplicació, on es detallen més detalladament les funcionalitats d'aquesta.

Casos d'ús de gestió de les bases de dades

Aquest cas d'ús ens ha de permetre gestionar les tres bases de dades del sistema (la que està en producció, la seva còpia i la versió antiga). Un cop implementada l'aplicació podem afirmar que es satisfan els requeriments corresponents i que podem gestionar les diferents bases de dades. Això implica que es pot seleccionar una base de dades per a actualitzar-la, fer-ne una còpia i canviar els rols jugats per les diferents bases de dades. També es poden configurar els paràmetres de connexió de les diferents bases de dades.

S'ha comprovat com la operació de realitzar la còpia entre bases de dades es fa dins d'un marge de temps raonable ja que triga tant sols alguns minuts a realitzar-se. L'aplicació indica de forma molt clara quin és el paper jugat en cada moment per cada una de les bases de dades ja que quan cliquem a "Change Version" aquestes canvien de posició en el sentit horari tot i prenent un nou rol. Un cop cliquem a "Use current version" accedim a l'aplicació general en la qual es carreguen els diferents plug-ins.

Casos d'ús de gestió d'organismes

En aquest cas d'ús s'han de poder gestionar els diferents organismes de la base de dades, per tant cal poder visualitzar-los, crear-los, eliminar-los i modificar-ne els atributs. S'han implementat totes aquestes funcionalitats i l'aplicació té el comportament esperat.

Casos d'ús de gestió de cromosomes

Aquest cas d'ús ens permet editar els cromosomes dels diferents organismes. L'aplicació implementada ens permet crear-los, eliminar-los i modificar-los correctament.

Casos d'ús de gestió de tracks

L'objectiu d'aquest conjunt de casos d'ús és poder gestionar els diferents Tracks amb els seus TrackViews corresponents. Podem editar aquests trackViews tant si es tracten de FeatureViews com si es tracten de ImageViews. L'aplicació satisfà de forma adequada aquestes funcionalitats.

Casos d'ús de la gestió de dades existents

A part de gestionar informació comuna a tots els plug-ins com són els organismes, els cromosomes i els tracks, l'aplicacació és capaç de gestionar correctament la informació específica del plug-in de Genbank. Així doncs podem visualitzar si les dades dels diferents cromosomes es troben actualitzades o desfasades respecte el servidor. Mitjançant icones se'ns mostra quines configuracions ens manquen per realitzar i l'aplicació ens permet accedir als diàlegs que ens permeten editar aquests paràmetres de configuració. Totes aquestes configuracions es guarden en fitxers XML que es generen i es tracten correctament.

L'actualització de les dades es fa de forma correcta, tot i que una mica lenta (els cromosomes més llargs poden tardar fins a 30 minuts a actualitzar-se), però cal remarcar que això no és necessàriament un problema doncs la actualització es realitza off-line.

7 Conclusions

Quan tot just fa un any vaig iniciar aquest projecte aviat vaig veure que hi havia moltes coses que es podien fer per millorar el rendiment de la base de dades, tot i així no sabia si seria capaç d'establir una estratègia clara de treball per assolir els objectius. Després de la feina feta estic satisfet amb el resultat obtingut doncs crec que aquests s'han complert satisfactòriament. A continuació es valorarà l'acompliment dels objectius fixats en el punt 3.3 de la memòria:

- **Implementar una nova base de dades per a GenExp minimitzant els temps de resposta**
  Objectiu assolit: S'ha redissenyat i implementat la nova base de dades de Genexp de manera que els temps de resposta s'adeqüin a les necessitats del navegador. A més, amb el nou model de dades s'ha guanyat la escalabilitat necessària per poder adminsitrar l'elevat nombre de dades.
- **Dissenyar una aplicació que permeti gestionar les dades de la base de dades**
  Objectiu assolit: S'ha dissenyat i implementat una aplicació que ens permet gestionar les dades emmagatzemades a la base de dades així com incorporar-ne de noves. Tot i que l'aplicació es pot considerar una mica lenta, gràcies a ella assegurem la integritat de la base de dades.
- **Definir una arquitectura de servidors**
  Objectiu assolit: S'ha dissenyat una estructura de servidors plausible per a la base de dades que doni resposta a un possible nombre elevat de peticions.

Un altre punt a destacar és el fet de que no s'ha materialitzat cap risc i, per tant, no ha calgut aplicar mesures de resposta com tornar enrere en el procés de desenvolupament. Aquest fet ha estat degut a que s'ha dedicat molt d'esforç a l'etapa de disseny ja que s'ha considerat que és la més important del procés de realització del projecte.

M'agradaria també definir algunes coses que es podrien fer per continuar amb la feina feta en aquest projecte. Per començar, una opció seria analitzar els diferents SGBD's d'aquí un temps i replantejar-se la seva utilització ja que gràcies a noves funcionalitats desenvolupades podria ser interessant canviar de SGBD i, per tant, caldria canviar el disseny físic de la base de dades. Una altra feina que es podria fer és definir de forma més acurada una arquitectura de servidors en el cas de que es materialitzi la possibilitat de que hi hagi una alta demanda de dades. Finalment, una altra feina a fer seria desenvolupar més plug-ins que captin informació genòmica de les fonts disponibles a Internet per a poder-les emmagatzemar a la base de dades.

Per tant, estic satisfet de poder afirmar que aquest projecte es tracta d'un treball reeixit i que, a més, m'ha permès aprendre molt. Gràcies a ell he pogut consolidar els meus coneixements sobre disseny de bases de dades, així com aprendre molts conceptes relacionats amb la bioinformàtica. A més també he pogut consolidar els meus coneixements de programació en Java que es tracta d'un camp del que sempre se'n pot aprendre més.

Vull acabar valorant com a molt positiva la meva participació en el grup de recerca que està desenvolupant el portal Genomport, del qual el meu PFC n'és tant sols una petita peça. Ha estat una experiència que m'ha permès posar en practica molts dels coneixements apresos a la carrera i, a més, aprendre molt.

8 Fonts i referències

Durant la realització d'aquest projecte s'han consultat les següents fonts d'informació:

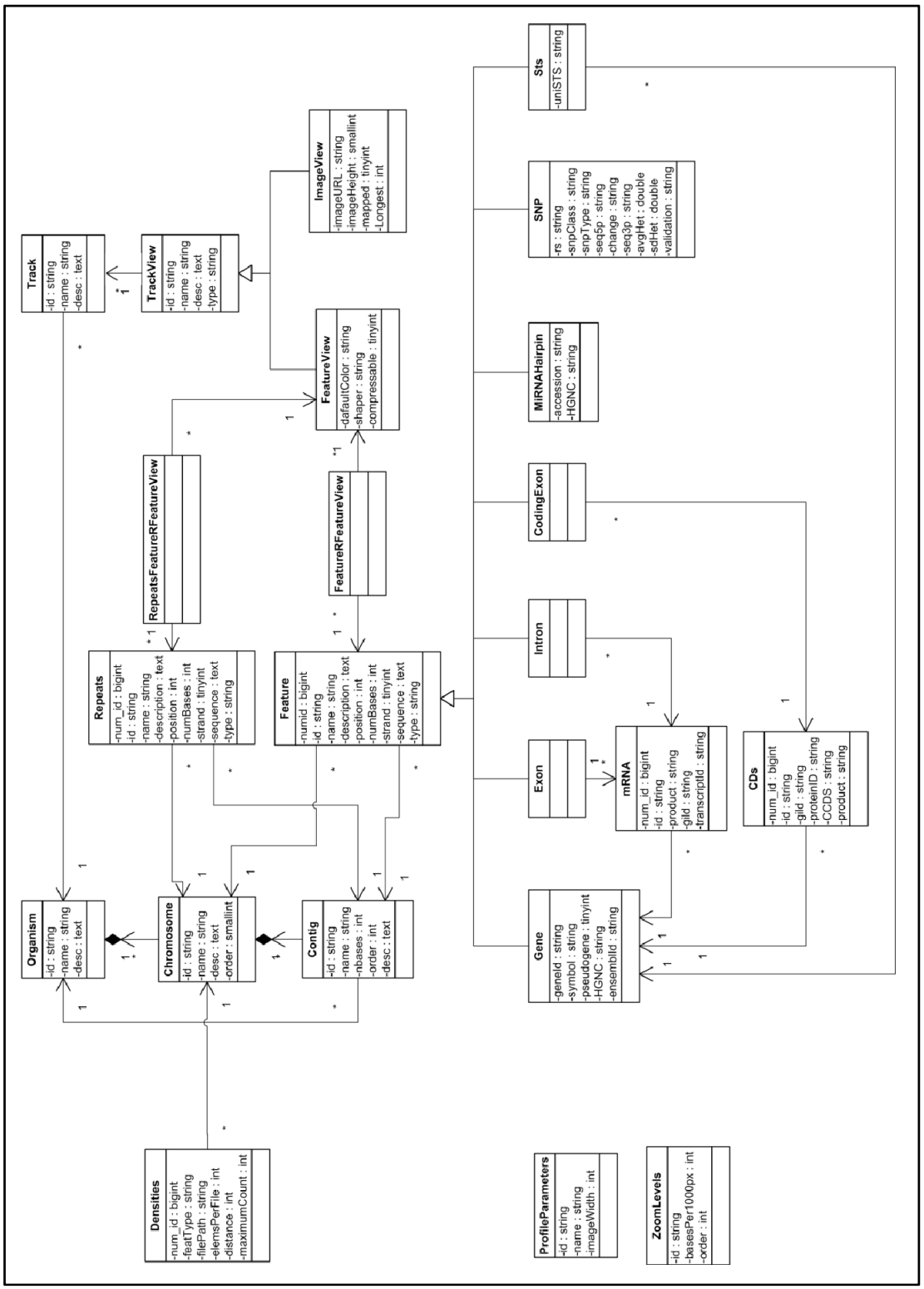
- Schwartz, Baron. *High Performance MySQL*. 2a edició
  EUA: O'Reilly Media, 2008
- Dyer, Russell J. T. *MySQL in a nutshell.* 1a edició
  EUA: O'Reilly, 2008
- Dubois, Paul. *MySQL Cookbook.* 1a edició
  Cambridge: O'Reilly, 2006
- Welling, Luke. *Mysql tutorial.* 1a edició
  EUA: Sams Publishing, 2004
- Robinson, Tara Rodden. *Genetics For Dummies.* 1a edició
  EUA: For Dummies, 2005
- Welling, Luke. *Mysql tutorial.* 1a edició
  EUA: Sams Publishing, 2004
- Abelló, Alberto. *Database design and administration.* 1a edició
  Barcelona: Edicions UPC, 2008
- MySQL and PostgreSQL Compared [En línia] PHP Builder
  <http://www.phpbuilder.com/columns/tim20000705.php3?page=1>
  [Cons. 17 setembre 2008]

Annex 1: Glossari de termes

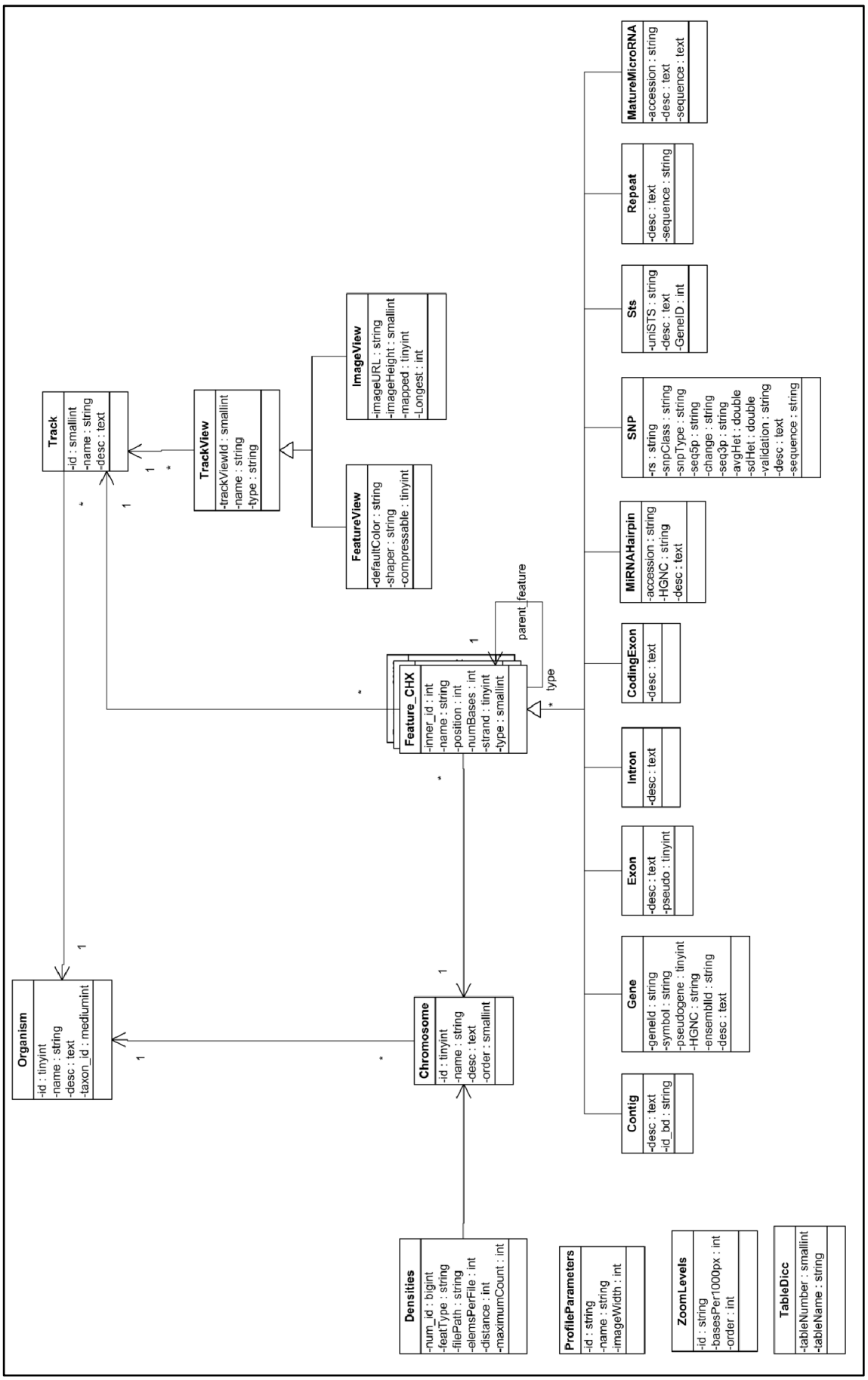
- **ADN:** L'àcid desoxiribonucleic (ADN o DNA) és un àcid nucleic que conté les instruccions genètiques utilitzades en el desenvolupament i funcionament de tots els éssers vius coneguts, així com en alguns virus. La funció principal de les molècules d'ADN és l'emmagatzemament d'informació a llarg termini.
- **AJAX:** AJAX són les sigles de Asynchronous Javascript And Xml, (JavaScript asíncron i XML), un conjunt de tecnologies que permeten actualitzar continguts web sense haver de tornar a carregar la pàgina.
- **Base:** Cada una de les molècules que formen la seqüència d'ADN.
- **Biojava:** El projecte Biojava és un projecte de codi obert que pretén proveir Java d'eines per processar dades biològiques.
- **B-Tree:** És una estructura de dades en forma d'arbre que manté les dades ordenades per a poder accedir-hi en un temps d'accés reduït.
- **Contig:** Conjunt de bases que s'han pogut seqüenciar de forma consecutiva.
- **Exó:** Part d'un gen que s'utilitza per codificar una proteïna.
- **Feature:** Cada una de les característiques de les que guardem informació, es pot tractar d'un gen determinat, d'un contig, d'un SNP...
- **Feature View:** Tipus de tracks que el visor mostra a través de codi HTML, no contenen massa característiques (fins a 30.000).
- **Gen:** Un gen és una seqüència linial bases que és essencial per a una funció específica, ja sigui en el desenvolupament de l'ésser o en el manteniment d'una funció fisiològica normal. És considerat com la unitat d'emmagatzemament d'informació i unitat d'herència al transmetre aquesta informació a la descendència.
- **Genbank:** És la base de dades de seqüències genètiques del NIH (National Institutes of Health d'Estats Units).
- **Genexp:** És el visualitzador genòmic que està essent desenvolupat per Bernat Gel dins del grup de recerca d'Algorísmia i Genètica de la UPC. La seva base de dades és la que pretén optimitzar aquest PFC.
- **Genomport:** És un portal web (http://www.genomport.cat) que pretén donar resposta a la necessitat d'eines acadèmiques en l'àmbit de la biomedicina. També es vol donar suport a la recerca en aquest àmbit oferint eines que facilitin el treball dels investigadors. Finalment, també es vol convertir en un punt de trobada per a aquelles persones, com metges, malalts i investigador, sobre un tema concret.
- **Image View:** Tipus de track que el visor mostra a partir de la generació d'una imatge, els fem servir per a aquells tracks que tenen moltes característiques.
- **Intró:** Part d'un gen que es troba entre dos exons.
- **JSON:** És l'acrònim de "JavaScript Object Notation", és un format lleuger per al intercanvi de dades.
- **MyISAM:** MyISAM és el sistema d'emmagatzemament per defecte del sistema gestor de base de dades MySQL.
- **MySQL:** MySQL és un sistema de gestió de bases de dades relaciona multi-fil , multiusuari, que usa el llenguatge SQL (Structured Query Language).
- **NCBI:** El Centre Nacional per a la Infomació Biotecnològica (National Center for Biotechnology Information) forma part de la National Library of Medicine (Biblioteca Nacional de Medicina) dels Estats Units, una branca dels National Institutes of Health (NIH) (Instituts Nacionals de Salut). Està localitzat a Bethesda, Maryland, i va ser fundat el 4 de novembre de 1988 amb l'objectiu de ser una font d'informació fonamental en el camp de la biologia molecular.

- **Perl:** És un llenguatge de programació dissenyat per Larry Wall. El llenguatge de programació Perl fou alliberat al 18 de desembre de 1987 per Larry Wall. Perl s'inspira en els llenguatges awk, C, sed, shell entre d'altres.
- **PHP:** És un llenguatge de programació interpretat utilitzat per a generar pàgines web dinàmiques. S'executa a la banda del servidor, per tant al navegador web la pàgina ja li arriba en format HTML i no podem veure'n el codi php.
- **PostgreSQL:** És un programari lliure que implementa un sistema de gestió de bases de dades objecte-relacional.
- **Repeat:** Seqüència de bases repetides de forma consecutiva.
- **SNP:** Sigles de *single nucleotyde polymorphysms*, (polimorfismes d'un sol nucleòtid), són polimorfismes d'un gen que ocorren per variació en un sol nucleòtid de la seqüència d'ADN. Els SNP contribueixen a les diferències entre els individus.
- **STS:** Sigles de *sequence-tagged site* (Lloc de seqüència etiquetada). Es tracta d'una curta (de 200 a 500 bases) seqüència de ADN que té una única ocurrència en el genoma i la posició de la qual és coneguda.
- **Track:** Cada un dels conjunts d'informació que podem mostrar. Per exemple tenim el track dels gens, el dels snps el dels Introns i Exons... Cada track es "pinta" en el visor com un feix horitzontal sobre el que es mostren les característiques.
- **UTR:** Són trossos de codi que marquen el inici (*UTR 5'*) i el final (*UTR 3'*) de la seqüència codificant
- **XML:** Sigles de eXtensible Markup Language (*llenguatge de marques extensible*), és un metallenguatge extensible, d'etiquetes, desenvolupat pel World Wide Web Consortium (W3C). És una simplificació i adaptació de l'experimentat SGML, i permet definir la gramàtica de llenguatges específics.

Annex 2: Ampliacions dels models conceptuals



Model conceptual previ a l'optimització



Model conceptual posterior a l'optimització

Annex 3: Descripció dels tipus de dades usats

Organism			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
id	1-40	UNSIGNED TINYINT	Sí
name	String (50)	VARCHAR(50)	No
desc	String	TEXT	No
taxonId	1-600.000	UNSIGNED MEDIUMINT	No

Chromosome			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
id	1-1.600	UNSIGNED SMALLINT	Sí
orgId	1-40	UNSIGNED TINYINT	No
name	String (50)	VARCHAR(50)	No
desc	String	TEXT	No
order	1-70	UNSIGNED TINYINT	No

Track			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
id	1-400	UNSIGNED SMALLINT	Sí
orgId	1-40	UNSIGNED TINYINT	No
name	String (50)	VARCHAR(50)	No
desc	String	TEXT	No

TrackView			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
trackViewId	1-800	UNSIGNED SMALLINT	Sí
trackId	1-400	UNSIGNED SMALLINT	No
name	String (50)	VARCHAR(50)	No
desc	String	TEXT	No
type	String(20)	VARCHAR(20)	No

FeatureView			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
trackViewId	1-800	UNSIGNED SMALLINT	Sí
defaultColor	String (100)	VARCHAR(100)	No
shaper	String (100)	VARCHAR(100)	No
compressable	Booleà	TINYINT(1)	No

ImageView			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
trackViewId	1-800	UNSIGNED SMALLINT	Sí
imageURL	String (100)	VARCHAR(100)	No
imageHeight	1-1.000	UNSIGNED SMALLINT	No
mapped	Booleà	TINYINT(1)	No
longest	1-200.000.000	UNSIGNED INT	No

Feature_CHX			
Atribut	**Rang de Valors**	**Tipus de dades**	**Clau primària**
innerId	1-1.000.000.000	UNSIGNED INTEGER	Sí
name	String (100)	VARCHAR(100)	No
position	1-500.000.000	UNSIGNED INTEGER	No
numBases	1-500.000.000	UNSIGNED INTEGER	No
strand	Booleà	TINYINT(1)	No
type	1-20	UNSIGNED TINYINT	No
trackId	1-400	UNSIGNED SMALLINT	No
parentFeature	1-1.000.000.000	UNSIGNED INTEGER	No

Annex 4: Sentències SQL

En aquest annex hi trobem les sentències SQL per crear les taules de la base de dades optimitzada.

```
--
-- Estructura de la taula `chromosome`
--

CREATE TABLE IF NOT EXISTS `chromosome` (
  `id` smallint(5) unsigned NOT NULL auto_increment,
  `orgId` tinyint(3) unsigned NOT NULL,
  `name` varchar(100) NOT NULL,
  `desc` text NOT NULL,
  `order` tinyint(4) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=25 ;


-- --------------------------------------------------------

--
-- Estructura de la taula `contig`
--

CREATE TABLE IF NOT EXISTS `contig` (
  `chrId` smallint(5) unsigned NOT NULL,
  `featureId` int(11) unsigned NOT NULL,
  `desc` text NOT NULL,
  PRIMARY KEY  (`chrId`,`featureId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `densities`
--

CREATE TABLE IF NOT EXISTS `densities` (
  `num_id` int(11) NOT NULL auto_increment,
  `featType` varchar(100) NOT NULL default '',
  `ChromosomeID` varchar(100) NOT NULL default '',
  `filePath` varchar(255) NOT NULL default '',
  `elemsPerFile` int(11) NOT NULL default '0',
  `distance` int(11) NOT NULL default '0',
  `maximumCount` int(11) NOT NULL default '0',
  PRIMARY KEY  (`num_id`),
  KEY `featType` (`featType`),
  KEY `ChromosomeID` (`ChromosomeID`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=69 ;

-- --------------------------------------------------------

--
-- Estructura de la taula `exon`
--

CREATE TABLE IF NOT EXISTS `exon` (
  `chrId` smallint(5) unsigned NOT NULL,
  `featureId` int(11) unsigned NOT NULL,
  `pseudo` tinyint(1) NOT NULL,
  `desc` text,
  PRIMARY KEY  (`chrId`,`featureId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
```

```
--
-- Estructura de la taula `featureview`
--

CREATE TABLE IF NOT EXISTS `featureview` (
  `trackViewId` smallint(5) unsigned NOT NULL,
  `defaultColor` varchar(100) NOT NULL,
  `shaper` varchar(100) NOT NULL,
  `compressable` tinyint(1) NOT NULL,
  PRIMARY KEY  (`trackViewId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `feature_chr1`
--

CREATE TABLE IF NOT EXISTS `feature_chr1` (
  `innerId` int(11) unsigned NOT NULL auto_increment,
  `name` varchar(100) NOT NULL,
  `position` int(10) unsigned NOT NULL,
  `numBases` int(11) NOT NULL,
  `strand` tinyint(1) NOT NULL,
  `type` smallint(6) NOT NULL,
  `trackId` smallint(4) unsigned NOT NULL,
  `parentFeature` int(11) unsigned NOT NULL,
  PRIMARY KEY  (`innerId`),
  KEY `positionTrack` (`position`,`trackId`),
  KEY `trackId` (`trackId`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `gene`
--

CREATE TABLE IF NOT EXISTS `gene` (
  `chrId` smallint(5) unsigned NOT NULL,
  `featureId` int(11) unsigned NOT NULL,
  `geneId` varchar(50) NOT NULL,
  `symbol` varchar(50) NOT NULL,
  `pseudogene` tinyint(1) NOT NULL default '0',
  `HGNC` varchar(50) default NULL,
  `desc` text,
  `otherXRef` text NOT NULL,
  `mrnaList` text,
  PRIMARY KEY  (`chrId`,`featureId`),
  KEY `SYMBOL` (`symbol`,`chrId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `imageview`
--

CREATE TABLE IF NOT EXISTS `imageview` (
  `trackViewId` smallint(5) unsigned NOT NULL,
  `imageURL` varchar(100) NOT NULL,
  `imageHeight` smallint(6) NOT NULL,
  `mapped` tinyint(1) NOT NULL,
  `longest` int(11) NOT NULL,
  PRIMARY KEY  (`trackViewId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
```

```
--
-- Estructura de la taula `intron`
--

CREATE TABLE IF NOT EXISTS `intron` (
  `chrId` smallint(5) unsigned NOT NULL,
  `featureId` int(11) unsigned NOT NULL,
  `desc` text,
  PRIMARY KEY  (`chrId`,`featureId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `organism`
--

CREATE TABLE IF NOT EXISTS `organism` (
  `id` tinyint(3) unsigned NOT NULL auto_increment,
  `name` varchar(50) NOT NULL,
  `desc` text,
  `taxonId` mediumint(9) default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;

-- --------------------------------------------------------
--
-- Estructura de la taula `profileparameters`
--

CREATE TABLE IF NOT EXISTS `profileparameters` (
  `id` varchar(100) NOT NULL default '',
  `name` varchar(255) NOT NULL default '',
  `imageWidth` int(11) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `ProfileParameters`
--

CREATE TABLE IF NOT EXISTS `ProfileParameters` (
  `id` varchar(100) NOT NULL default '',
  `name` varchar(255) NOT NULL default '',
  `imageWidth` int(11) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `repeat`
--

CREATE TABLE IF NOT EXISTS `repeat` (
  `chrId` tinyint(3) unsigned NOT NULL,
  `featureId` int(10) unsigned NOT NULL,
  `patternLength` int(3) unsigned NOT NULL,
  `desc` text,
  `pattern` text,
  `nRep` tinyint(3) unsigned NOT NULL,
  PRIMARY KEY  (`chrId`,`featureId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------
```

```
--
-- Estructura de la taula `sts`
--

CREATE TABLE IF NOT EXISTS `sts` (
  `chrId` smallint(5) unsigned NOT NULL,
  `featureId` int(11) unsigned NOT NULL,
  `uniSTS` varchar(100) NOT NULL,
  `desc` text,
  `geneSymbol` varchar(50) NOT NULL default '0',
  PRIMARY KEY  (`chrId`,`featureId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `tabledict`
--

CREATE TABLE IF NOT EXISTS `tabledict` (
  `tableNumber` smallint(6) unsigned NOT NULL auto_increment,
  `tableName` varchar(20) NOT NULL,
  PRIMARY KEY  (`tableNumber`,`tableName`),
  UNIQUE KEY `tableName` (`tableName`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=8 ;

-- --------------------------------------------------------

--
-- Estructura de la taula `track`
--

CREATE TABLE IF NOT EXISTS `track` (
  `id` smallint(5) unsigned NOT NULL auto_increment,
  `name` varchar(100) NOT NULL,
  `desc` text,
  `orgId` tinyint(3) unsigned NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=19 ;

-- --------------------------------------------------------

--
-- Estructura de la taula `trackview`
--

CREATE TABLE IF NOT EXISTS `trackview` (
  `trackId` smallint(5) unsigned NOT NULL,
  `trackViewId` smallint(5) unsigned NOT NULL auto_increment,
  `name` varchar(100) NOT NULL,
  `type` varchar(20) NOT NULL,
  `desc` text NOT NULL,
  PRIMARY KEY  (`trackViewId`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=18 ;

-- --------------------------------------------------------

--
-- Estructura de la taula `utr`
--

CREATE TABLE IF NOT EXISTS `utr` (
  `chrId` smallint(5) unsigned NOT NULL,
  `featureId` int(11) unsigned NOT NULL,
  `desc` text NOT NULL,
  `type` varchar(2) NOT NULL,
  PRIMARY KEY  (`chrId`,`featureId`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
--
-- Estructura de la taula `zoomlevels`
--

CREATE TABLE IF NOT EXISTS `zoomlevels` (
  `id` char(100) NOT NULL default '',
  `basesPer1000px` int(11) NOT NULL default '0',
  `order` mediumint(9) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--
-- Estructura de la taula `ZoomLevels`
--

CREATE TABLE IF NOT EXISTS `ZoomLevels` (
  `id` char(100) NOT NULL default '',
  `basesPer1000px` int(11) NOT NULL default '0',
  `order` mediumint(9) NOT NULL default '0',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Annex 5: Manual d'usuari de l'aplicació de control

Aquest annex conté el manual d'usuari de l'aplicació de control de dades.

Introducció

L'aplicació de control de dades de Genexp ens permet administrar les dades genòmiques que es guarden a la base de dades de Genexp i que seran mostrades pel navegador. Així doncs, el programa ens permet gestionar els diferents organismes, cromosomes i tracks així com les pròpies característiques genòmiques emmagatzemades.

Inici de l'aplicació

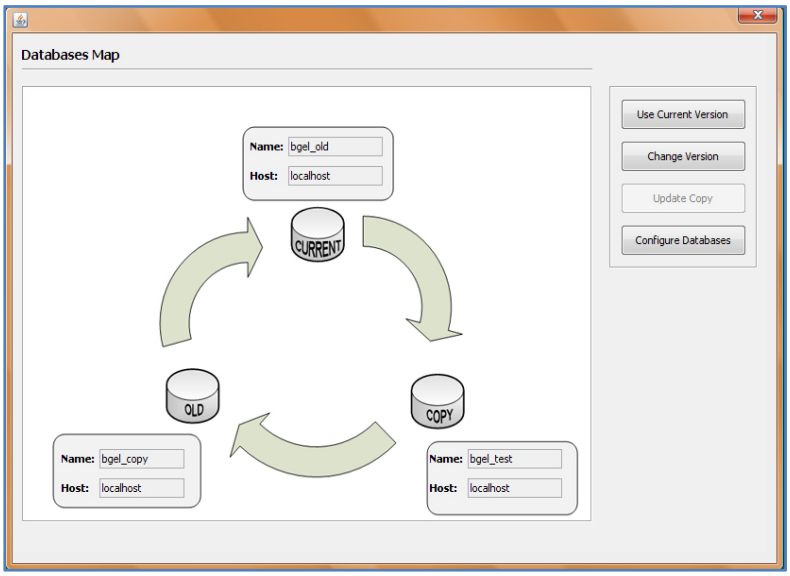
Per iniciar l'aplicació hem d'usar l'arxiu *run.bat* si estem a Windows o el *run.sh* si estem a Linux. A continuació se'ns permet administrar les diferents bases de dades.

Gestió de les bases de dades

L'aplicació ens permet gestionar quin paper juguen les tres bases de dades assignades a l'aplicació. Aquests rols són els següents:

- Base de dades actual
- Còpia de la base de dades actual
- Base de dades antiga (congelada)

A continuació veiem el diàleg que ens ho permet gestionar:



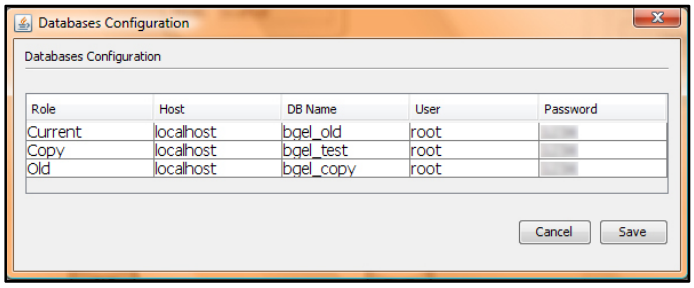
A la figura hi veiem quin paper juguen les diferents bases de dades identificades pel seu nom i host.

Si cliquem a "Change Version" el que fem és canviar els rols jugats per les bases de dades de la següent manera:

- La versió antiga passa a ser l'actual, però abans s'hi carreguen les dades actuals
- La versió actual passa a ser la còpia de l'actual
- La versió de còpia passa ser la versió antiga (o congelada)

Clicant a "Update Copy" el que fem és assegurar que la versió de còpia i la versió actual contenen les mateixes dades.

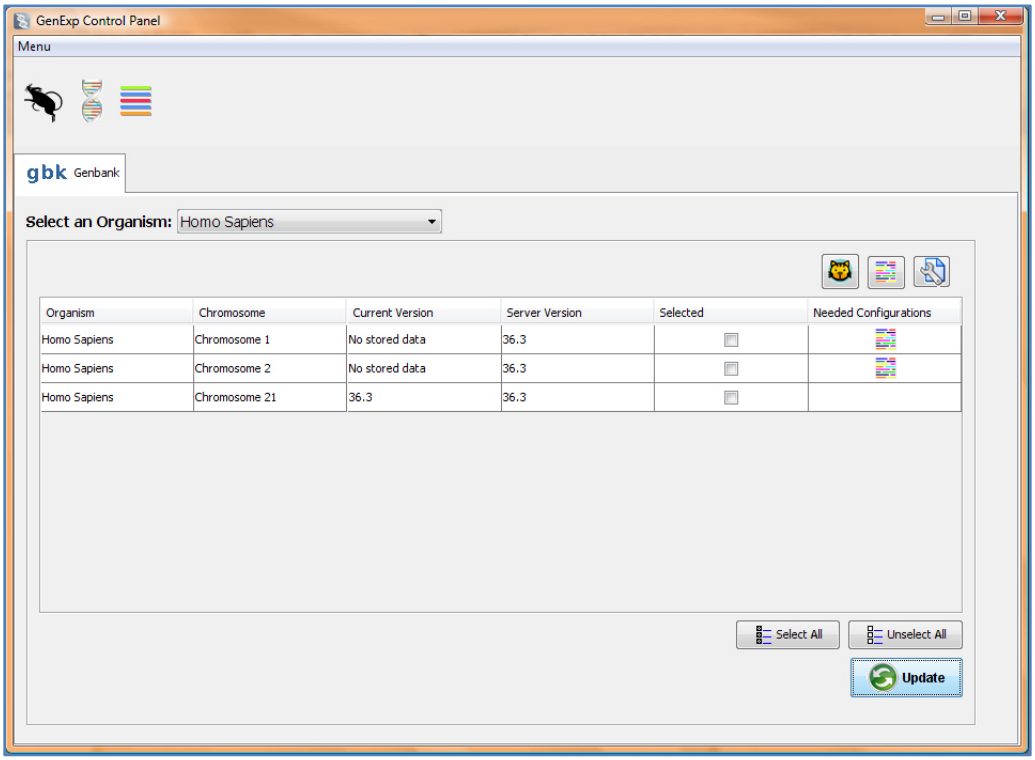
Si accedim a "Configure Databases" el que fem és accedir a un diàleg que ens permet editar els paràmetres de connexió de les diferents bases de dades. Ho veiem a la següent figura:



Si cliquem a "Use Current Version" accediríem al diàleg principal de l'aplicació que ens permet administrar les dades de la base de dades actual.

Gestió de les dades genòmiques

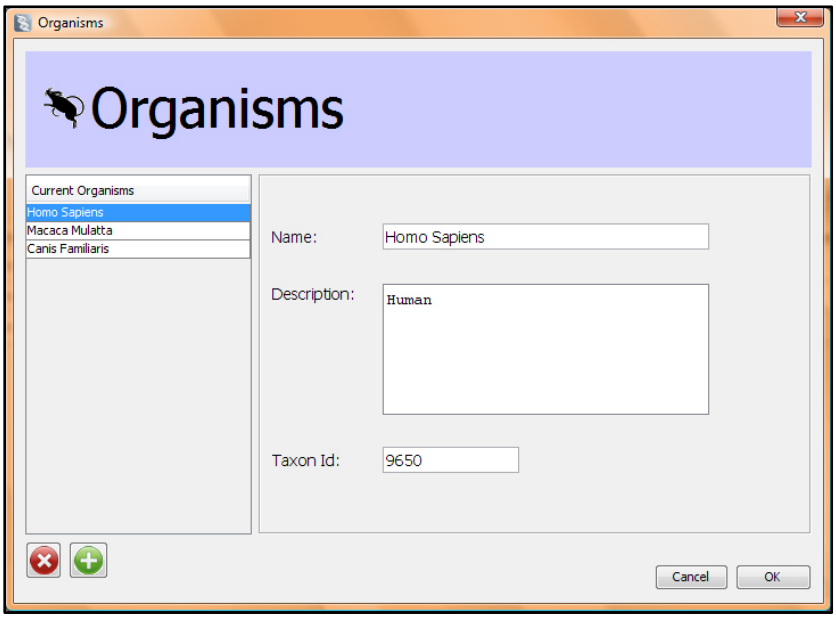
Quan cliquem a "Use Current Version" ens apareix el següent diàleg principal:



A la part superior hi ha les icones que ens permeten accedir als diàlegs de gestió d'organismes, cromosomes i tracks. Mentre que a la part inferior hi trobem les diferents pestanyes dels plug-ins existents.

Gestió dels organismes

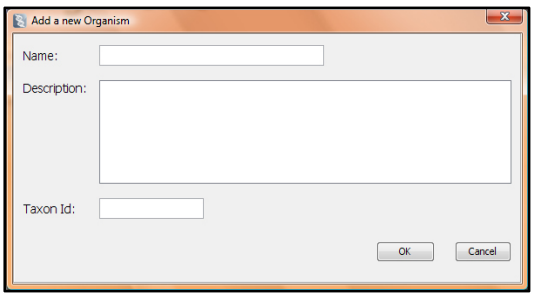
Si cliquem a aquesta icona  , accedim al diàleg que ens permet gestionar els organismes i que veiem a continuació.



Quan seleccionem un organisme de la llista podem veure els seus atributs i els podem modificar, els canvis es guardaran automàticament.

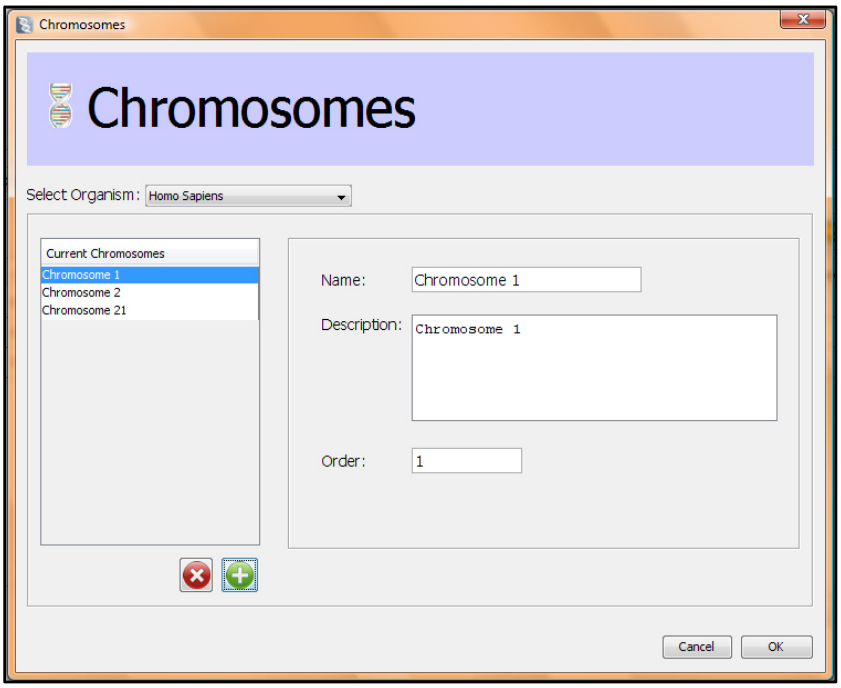
Si volem eliminar un organisme hem de clicar a aquesta icona: 

Si el que volem és afegir un organisme hem de clicar a  . A continuació se'ns obre el següent diàleg que ens permet editar els atributs del nou organisme:

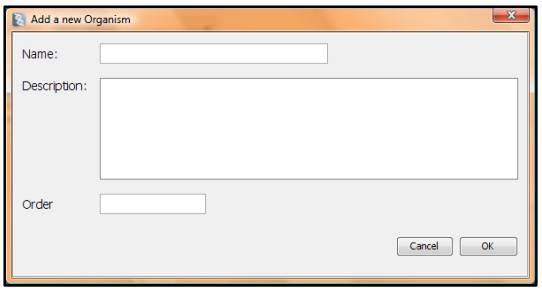


Gestió de cromosomes

Si en el diàleg principal cliquem a  accedim al diàleg que ens permet editar els cromosomes:

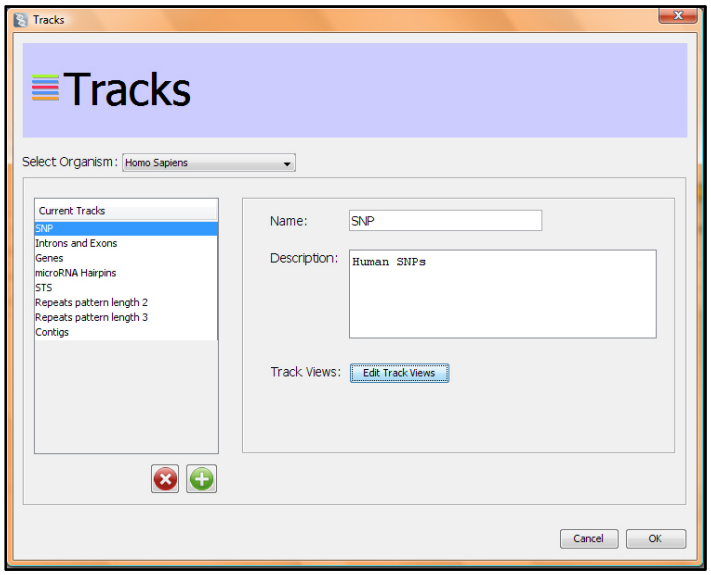


En aquest diàleg hi podem seleccionar l'organisme del que volem editar els cromosomes mitjançant el desplegable que es troba a la part superior. El funcionament és el mateix que per editar els organismes. Si cliquem a l'icona  , accedim al diàleg que ens permet crear un nou cromosoma:

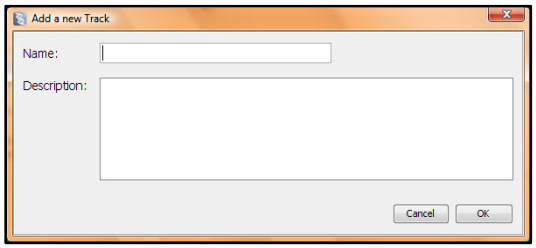


Gestió de tracks

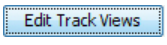
Si en el diàleg principal cliquem a  accedim al diàleg que ens permet editar els tracks:

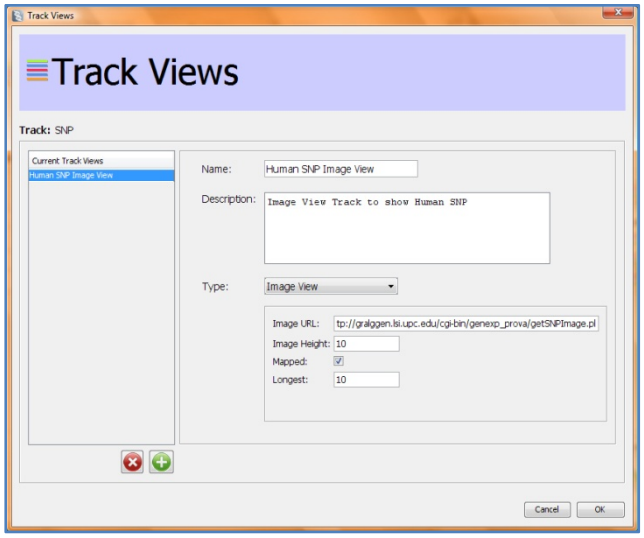


Mitjançant el desplegable de la part superior podem seleccionar l'organisme del que volem veure els tracks associats. Si cliquem a  accedim al diàleg que ens permet crear un nou track:



Gestió dels TrackViews d'un track

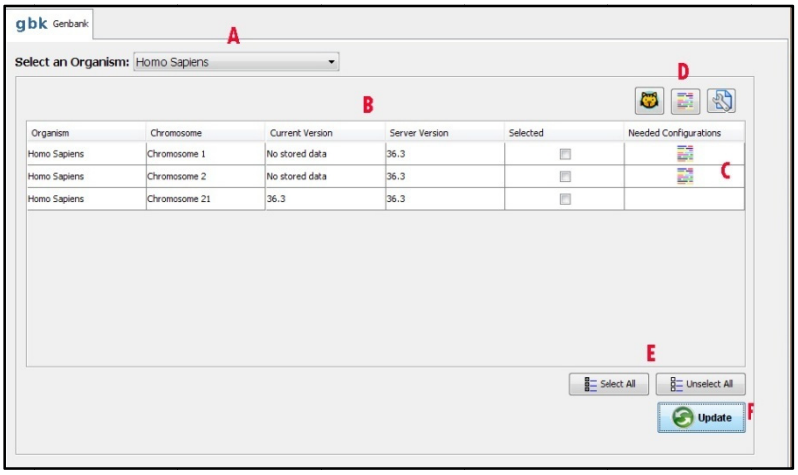
Si cliquem al botó  que apareix al diàleg de la configuració dels tracks podem editar els diferents TrackViews d'un track determinat:



Amb aquest diàleg podem eliminar TrackViews existents, crear-ne de nous, o modificar els existents.

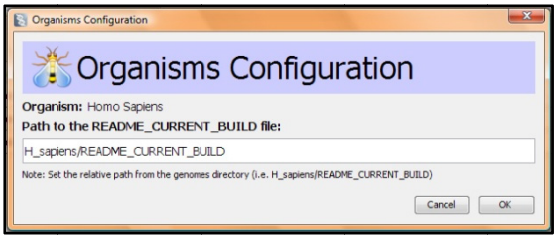
Plug-in de Genbank

A continuació veurem les diferents funcionalitats que ofereix el plug-in de Genbank:

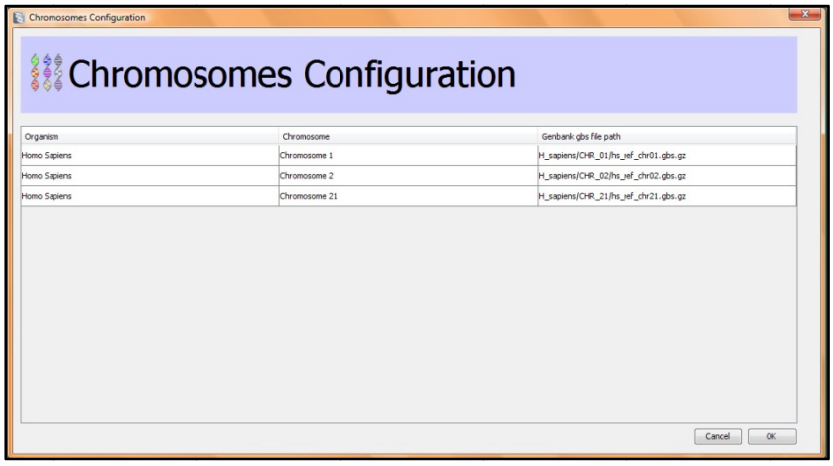


Mitjançant **A** podem seleccionar de quin organisme volem visualitzar l'estat de les dades genòmiques. A la taula **B** hi podem veure la informació dels diferents cromosomes. Concretament hi veiem l'organisme, el nom del cromosoma, la versió que hi ha guardada a la nostra base de dades i la versió que hi ha actualment a Genbank. A la columna **C** hi veiem les icones que ens indiquen quines configuracions manquen de realitzar. Amb les icones **D** accedim als diàlegs que ens permeten fer les configuracions corresponents.

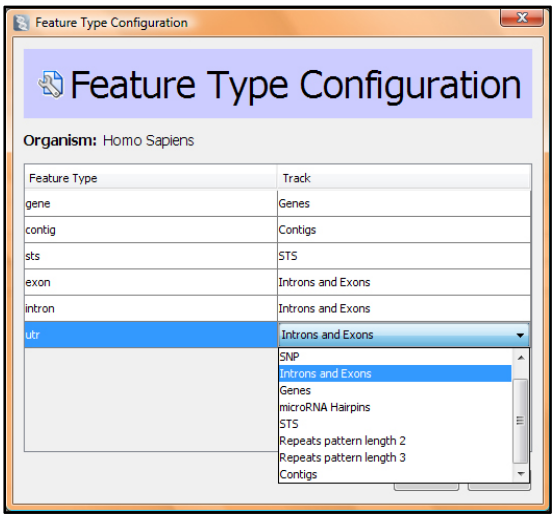
Si falta per configurar la URL que ens indica el fitxer on es troba la versió de l'organisme hem de clicar a la icona  per poder accedir al diàleg que ens permet introduir aquest paràmetre:



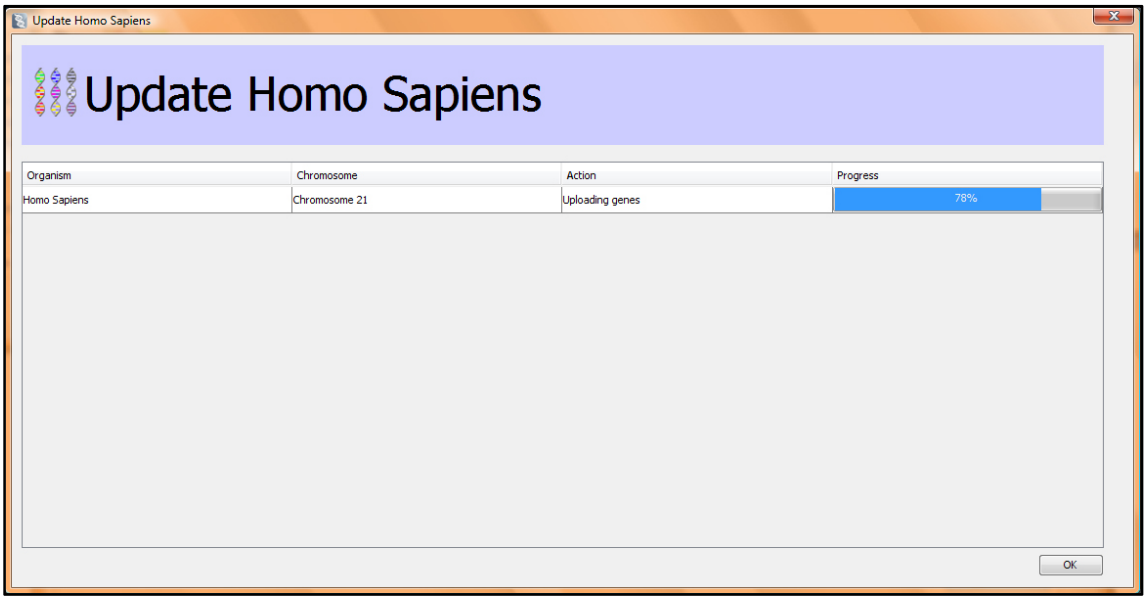
Si el que falta per configurar és la URL on es troba el fitxer gbs del cromosoma cal clicar a 



Finalment, si el que falta per configurar és l'assignació dels tipus de dades que ofereix Genbank amb els tracks creats cal clicar a . A continuació accedim al diàleg que ens permet fer aquesta assignació mitjançant la selecció del track desitjat en una llista desplegable.



Ja només ens queda seleccionar quins cromosomes volem actualitzar, ho podem fer manualment mitjançant el camp "Selected" de la taula o els botons **E**. Un cop els hem selecionat hem de clicar al botó **F** per a actualitzar-los. Quan ho fem, se'ns obre el següent diàleg:



En aquest diàleg se'ns monitoritza el procés d'actualització dels cromosomes. Cal notar que no podem parar aquest procés fins que hagi finalitzat.